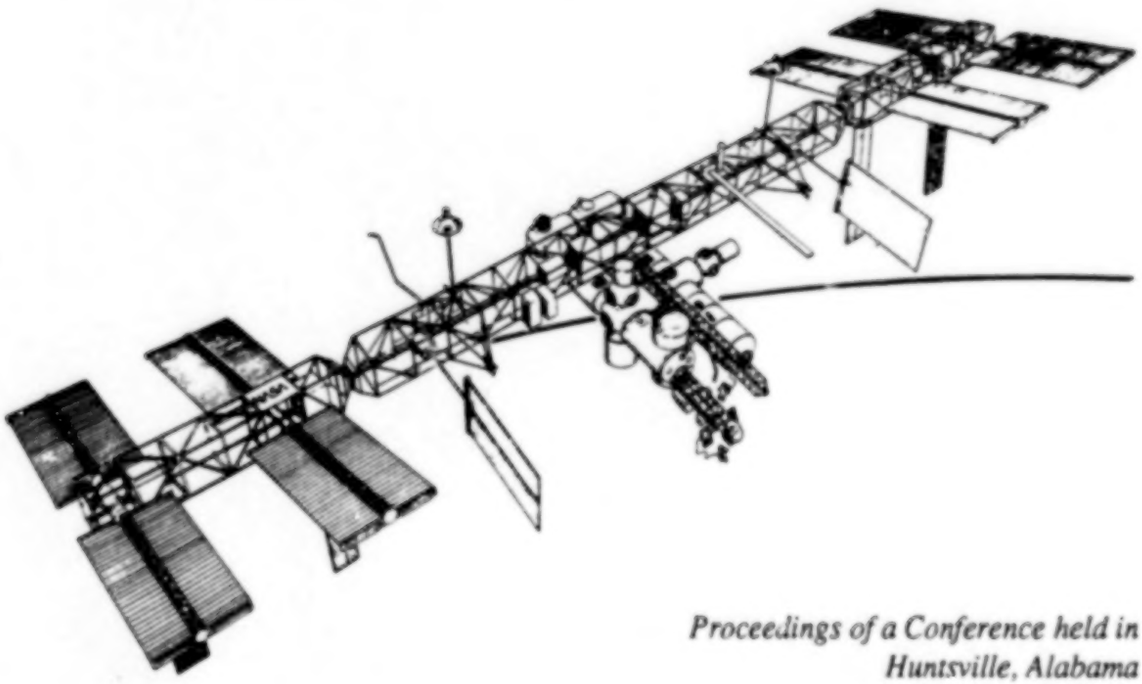
NASA Conference Publication 2492

Third Conference on Artificial Intelligence for Space Applications

Part II

N88-24188
--THRU--
N88-24197
Unclas
H1/61 0146548

(NASA-CP-2492-Pt-2) THIRD CONFERENCE ON ARTIFICIAL INTELLIGENCE FOR SPACE APPLICATIONS, PART 2 (NASA) 66 p CSCL 09B



Proceedings of a Conference held in Huntsville, Alabama November 2-3, 1987

Sponsored by:

The University Of Alabama In Huntsville

NASA

NASA Conference Publication 2492

Third Conference on Artificial Intelligence for Space Applications

Part II

Compiled by
J. S. Denton,
M. S. Freeman,
and M. Vereen

Proceedings of a conference sponsored by
The University of Alabama in Huntsville and
The National Aeronautics and Space Administration
and held in Huntsville, Alabama
November 2 -3, 1987

NASA
National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1988

CONFERENCE COORDINATOR

THOMAS S. DOLLMAN
INFORMATION AND ELECTRONIC SYSTEMS LABORATORY
MARSHALL SPACE FLIGHT CENTER

CONFERENCE CO-CHAIRMEN

JUDITH S. DENTON, MSFC
GARY L. WORKMAN, UAH

PROCEEDINGS COORDINATOR

MARY VEREEN

LOGISTICS AND ARRANGEMENTS COMMITTEE

DAVID WEEKS, MSFC
KAREN MACK, UAH

TECHNICAL AND PROGRAM COMMITTEE

MICHAEL FREEMAN, MSFC
JAMES JOHANNES, UAH

ROWLAND BURNS, MSFC
DAN HAYS, UAH
ELAINE HINMAN, MSFC
BERNARD SCHROER, UAH
CAROLINE WANG, MSFC

PUBLICATIONS COMMITTEE

LUSTER INGRAM, MSFC
BERNARD SCHROER, UAH

JEANETTE REISZ, MSFC
WILLIAM SELIG, MSFC

PUBLICITY COMMITTEE

JUDITH S. DENTON, MSFC
GARY L. WORKMAN, UAH

GERRY HIGGINS, MSFC
KAREN MACK, UAH

ADVISORS

JONATHAN HAUSSLER, MSFC
ELAINE HINMAN, MSFC
GABE WALLACE, MSFC

FOREWORD

This document contains the Proceedings of the Third Conference on Artificial Intelligence for Space Applications (CAISA), sponsored by the National Aeronautics and Space Administration's (NASA's) George C. Marshall Space Flight Center (MSFC) and the University of Alabama in Huntsville (UAH). There is widespread interest throughout the aerospace community in utilizing scientific and technical developments from the field of Artificial Intelligence (AI) to enhance our space program. For NASA, future activities in space will rely on the effective utilization of key AI components. The intent of this conference is to provide an opportunity for those groups and individuals who employ AI methods in space applications to identify common goals, to compare the effectiveness of the various approaches being investigated, and to discuss issues of general interest in the AI community. The Third CAISA brings together a diversity of scientific and engineering work and is intended to promote thoughtful discussion concerning the possibilities created by this work.

AI contains many components, some of which can be selectively applied to develop more competent, less demanding flight/ground systems. This is the message of the invited speakers at our keynote session. As the participants in this conference have recognized, there is no more fascinating - nor more potent - combination of technologies than is found in the use of Artificial Intelligence to support our exploration of space. The potential benefits to our society and all mankind are literally limitless. The presentations in our technical sessions discuss various aspects of this technology. The papers presented were selected through a careful review of the submitted abstracts by at least five members of the Technical Committee. The selected presentations, represented by the papers or abstracts herein, were organized into twenty-one technical sessions. Every effort was made to minimize the conflicts arising from parallel sessions. The broad range of topics presented is indicative of the interest in NASA's goals commonly found in the AI community.

This conference would not have been possible without the dedicated efforts of many people. First, I would like to thank the authors whose research and development efforts are presented here. Second, I thank the members of all the committees, the advisors, and the volunteers who planned and implemented the numerous activities which enable a conference such as this one. I thank the exhibitors for their efforts to develop and demonstrate tools for implementing many of the ideas discussed during these two days. And, finally, I thank the invited speakers and the many other people from NASA and UAH whose interest in Artificial Intelligence and space applications makes this conference both possible and meaningful.

Thomas Dollman

This document contains those Proceedings which were not incorporated in Part I.

PAGE ii INTENTIONALLY BLANK

TABLE OF CONTENTS

NEW TECHNOLOGIES FOR SPACE STATION AUTOMATION

MTK: An AI Tool for Model-Based Reasoning
W. K. Erickson and M. R. Rudokas 1

Integration of Symbolic and Algorithmic Hardware and Software for the Automation of Space Station Subsystems
H. Gregg, K. Healey, E. Hack and C. Wong 7

Connecting Remote Systems for Demonstration of Automation Technologies
R. M. Brown and R. Yee 15

Knowledge Based System Verification and Validation as Related to Automation of Space Station Subsystems: Rationale for a Knowledge Based System Lifecycle
K. Richardson and C. Wong 25

DESIGN DATA CAPTURE

TALOS: A Distributed Architecture for Intelligent Monitoring and Anomaly Diagnosis of the Hubble Space Telescope
Bryant G. Cruse 31

COMPUTER VISION

Orbital Navigation, Docking, and Obstacle Avoidance as a Form of Three Dimensional Model-Based Image Understanding
J. Beyer, C. Jacobus and R. Mitchell 37

NEURAL NETS

Genetic Algorithms for Adaptive Real-Time Control in Space
J. van der Ziip and A. Choudry 47

AUTOMATIC PROGRAMMING

Automatic Program Generation from Specifications Using Prolog
Alex Pelin and Paul Morrow 53

PRECEDING PAGE BLANK NOT FILMED

PAGE iv INTENTIONALLY BLANK

REAL-TIME APPLICATIONS

TES - A Modular Systems Approach
to Expert System Development for
Real-Time Space Applications
Ralph Cacace and Brenda England

N88-24189

MTK: An AI Tool For Model-Based Reasoning

William K. Erickson and Mary R. Rudokas
Systems Autonomy Demonstration Project Office
NASA Ames Research Center
Moffett Field, California, 94035

Abstract

A 1988 goal for the Systems Autonomy Demonstration Project Office of the NASA Ames Research Center is to apply model-based representation and reasoning techniques in an knowledge-based system that will provide monitoring, fault diagnosis, control, and trend analysis of the Space Station Thermal Control System (TCS). A number of issues raised during the development of the first prototype system inspired the design and construction of a model-based reasoning tool called MTK, which was used in the building of the second prototype. This paper outlines these issues, with examples from the thermal system to highlight the motivating factors behind them, followed by an overview of the capabilities of MTK, which was developed to address these issues in a generic fashion.

Background

The Systems Autonomy Program is a NASA Office of Aeronautics and Space Technology (OAST) sponsored program to develop and demonstrate the application of A.I. technology with a focus on Space Station [2][6]. As part of Ames' role as coordinating center for the Systems Autonomy Program, the Systems Autonomy Demonstration Project Office of the Information Sciences Division at NASA Ames Research Center is currently developing a knowledge-based system that will support monitoring, fault diagnosis, control, and trend analysis on the Space Station Thermal Testbed. The knowledge-based system, to be demonstrated in 1988, has been dubbed TEXSYS for Thermal Expert System. As part of the development of this system, two research prototypes were developed to evaluate approaches and highlight technical issues that would arise in the development of the full-fledged knowledge-based system. These issues, to be described later, were seen to be relevant to a large class of applications, and inspired the development of a generic model building and inferencing tool, which has been called MTK (Model Toolkit).

The Space Station Thermal Testbed

The Space Station Thermal Testbed, currently undergoing integration, test, and evaluation at the NASA Johnson Space Flight Center (JSC), is being used to evaluate the utility of a number of thermal technologies for use within the Space Station Thermal Control System (TCS) [4][5]. The Space Station TCS is used to collect excess heat that is generated by the electronics, experiments, and crew aboard the Space Station and transport it to external radiators, where it is radiated out into space.

The TEXSYS Prototypes

Work on the prototype system occurred in two phases, with phase I completed in June 1986 and phase II completed in February 1987. Domain expertise for both phases was acquired from the chief engineer on the thermal testbed and the engineer who developed a differential equation based mathematical model of the system.

The phase I prototype focused primarily on the development of a simplified causal model of the thermal system, performing static state simulation (deducing additional state information from partial information at a single moment in time) and limited fault diagnosis. The prototype was developed on a Symbolics 3670[1] using ZetaLisp[1] , KEE[2] version 2, and SimKit[2]. The basic architecture utilized KEE's object-oriented frame-based representation: frames with inheritance to represent structure, rules to capture behavior and fault diagnosis, and methods for control and bookkeeping. Two approaches for representing the topology of the system were informally tested: one technique utilized direct component links/pointers; the other used intermediate objects called PORTS to represent the interactions between components. While direct links yielded a simpler design, PORTS offered greater perspicuity and power by allowing for the representation of behavior at the boundaries of components. A simplified model of the thermal testbed was constructed, incorporating basic versions of the relevant physical laws and the domain expert's heuristic rules of thumb. A limited hierarchical representation capability allowed the prototype to model system-level as well as component-level behavior.

The phase II prototype was developed on a Symbolics 3670 using KEE version 3, Common Lisp, and MTK. KEE3 provided a number of advanced features upon which the Model Toolkit was built. KEEpictures provided a flexible graphics interface. KEEworlds provided a set of facilities for representing different hypothetical or temporal states of the same system in distinct "worlds". This allowed for representing the system as it changes state over time for dynamic simulation, or for hypothetical reasoning over different alternatives during fault diagnosis. KEEworlds includes an assumption-based truth maintenance system which can track assumed and derived beliefs and their associated justifications in various worlds. Model topology was represented using CONNECTIONS, an enhanced version of PORTS. Other enhancements in the second prototype included more detailed causal modeling of both nominal and faulty system behavior, a more complete model of the thermal testbed, and more detailed representations of the individual components. The availability of test results from ambient tests on the thermal testbed [3] provided information on the performance characteristics of the components, producing more accurate rules for modeling system behavior. Information on faults that occurred during the tests provided additional knowledge to enhance the fault diagnosis capabilities of the prototype. In addition, data dumps of sensor readings collected during the tests were used for simulated "test runs" of the model to verify the accuracy of its analysis of system behavior.

issues

A number of technical issues were raised during development of the phase I prototype that inspired the development of MTK as a generic development tool that could be used for other projects with similar model-based approaches. These issues are described in the next few sections.

Causal Modeling

The thermal system is made up of distinct, easily definable pieces of hardware that fit together and interact with one another in well-defined and limited ways. Pipes, valves, pumps, condensers, evaporators, and other components are completely modular, behave in a reasonably understood fashion, and interact with one another via a few defined connections. Coolant enters and leaves a component via recognized inlets and outlets. Heat is transferred into, out of, and within the system through pre-defined mechanisms associated with particular components: evaporators acquire heat, radiators release heat, and pipes transport coolant that carry heat. For these reasons, the thermal system is an ideal candidate for qualitative causal modeling, simulation, and fault diagnosis [1].

---

1 Trademark Symbolics Inc.

2 Trademark IntelliCorp.

Graphic Representation and Interface
Trying to understand or build the model of a system by directly examining or changing the values of attribute slots in a frame representation is awkward and slow. In addition, it is hard to get a high-level understanding of the system as a whole by tracing out the topology via attribute slots. A flexible graphics interface provides a useful approach for representing components and their interactions, as well as for actually building the model by allowing for the interactive creation of component instances and graphically defining how they interact with one another.

Hierarchical Structure
At a certain level of detail, many components within the thermal system can be viewed as "black boxes" with specified inputs and outputs. But at another level, it may be necessary to look inside the "black box" at the substructure within. Different levels of simulation and fault diagnosis will require different levels of hierarchical structure, from system to subsystem to component to subcomponent, with perhaps other levels in between.

Quantitative and Qualitative Information
Qualitative causal modeling provides a good high-level picture of what is happening with the thermal system, capturing much of the heuristic knowledge of the thermal engineer. But in addition, a large amount of quantitative information is available, together with physical laws that establish how these quantitative parameters relate to and interact with one another. Although each level of modelling can be used disjointly, more effective use can be made if both could be integrated in some fashion.

Parameter Uncertainty
In any modeled physical system, there is always a level of uncertainty concerning the actual value of parameters in the system. Sensor readings are only accurate to within some range; temperature sensors in the thermal system are only accurate to within 3 degrees. Performance characteristic curves of components are only approximations of how the component responds in the system, as determined from earlier test results of that component. It is also possible to represent some qualitative information as a numeric value with some amount of uncertainty associated with it.

Supporting and Conflicting Evidence
The value of a parameter within a system may be derivable from several different sources. A sensor may provide one value. Another value may be derived from heuristic performance characteristic curves. Yet another value may be derivable from first principles using physical laws. Each value may vary in level of uncertainty, and may support one another or, if based upon faulty assumptions, may conflict. For example, if a flow sensor is faulty, the value for flow derived from that sensor may conflict with other values for flow derived from rules of inference and alternate assumptions. A mechanism for evaluating supporting and conflicting evidence can provide useful support for fault diagnosis and integrating different levels of information.

MTK: The Model Toolkit

Originally developed as a replacement for the SimKit package used in the first prototype, MTK soon absorbed additional support utilities designed to address the issues raised above, and quickly became a major part of the design effort for the second phase prototype of the TEXSYS system.

Structures and Connections
Support for causal modeling in MTK is provided using structures to represent components and connections to represent the interactions between components. Structures are based upon KEE's

object oriented frame representation. Structures are used to represent discrete physical components, such as pipes, values, evaporators, and the like. In addition, structures can represent more abstract entities, such as the entire system, subsystems, or other functional composites that need to be represented. For each object, connections are defined which capture all the ways in which one component will interact with others.

Graphics Interface
In MTK, pictoral icons within a model viewport window are used to represent both components and their connections, with lines indicating links between connections. Component and connection icons can be edited in the final version to produce a graphic representation that closely approximates that of an engineering schematic.

Hierarchical Structures
In MTK, substructure is represented in a separate component viewport window that can be associated with any component icon. The viewport includes icons representing the connections of the top-level component represented, plus the icons of the subcomponents, their connections, and the links between these connections.

Parameters
Parameters represent the significant physical measurements important in describing a given system. In the thermal system these include temperature, pressure, flow rate, and heat. Parameters within MTK are represented as special objects with value, best value, state, and history attributes.

Value The value of a parameter is a quantitative amount, such as 72 degrees for a temperature. Values can be represented as a range, such as 40 plus or minus 5: (40 +- 5). Values may be assumed, or derived from other parameters and rules in the knowledge base. Since there can be different rules using different assumptions to deduce a value, there can be multiple values associated with a parameter, corresponding to different chains of inference.

Best Value The best value of a parameter is the system's best guess of the "real" value of a parameter, given all the assumed and derived values mentioned above. Whenever a new value is added to the knowledge base, a new best value is calculated by combining all the values according to a selected merging function. If the merging function detects a conflict between two values, a conflicting evidence resolution procedure is activated to evaluate the evidence in support of each value and attempt to determine which basic assumptions may be suspect. From this information further fault diagnosis can be performed.

State The state of a parameter is a qualitative symbolic value. Valid symbols could include LOW, NOMINAL, HIGH, NEGATIVE, ZERO, POSITIVE, or any other designation that is appropriate for this parameter.

History History is another qualitative parameter summarizing the recent time behavior of the parameter. Typical values could include STEADY, INCREASING, or DECREASING. Rules can be defined to deduce history from the current and recent past parameter values.

The ability to represent values as a range or a qualitative state is a useful mechanism for dealing with parameter uncertainty. The use of value, state, and history information within parameters allow for both quantitative and qualitative modelling of a system. Rules within the library knowledge base can be used to define quantitative values (with ranges) from qualitative information, and qualitative state from numeric values. The merging and conflicting evidence resolution functions associated with the best value are useful mechanisms in dealing with supporting and conflicting evidence. Together with KEEworlds and the truth maintenace system, MTK provides mechanisms that allow the system to evaluate and compare the basic

assumptions that justify conflicting conclusions, and with appropriate rules deduce which assumption is invalid, or at least reduce the possibilities down to a set of candidate suspect assumptions, from which separate hypothetical worlds can be generated to evaluate each alternative in turn.

Summary

The phase I development of a prototype system to support the Space Station Thermal Testbed raised a number of technical issues. These issues were recognized as relevant to a large class of problem domains associated with causal modeling. This has inspired and directed the development of a expert system development toolkit called MTK to address these issues, which has been used within the phase II prototype, and promises to provide useful support for future projects involving model-based representation and reasoning.

Acknowledgements

The authors would like to acknowledge the support of a number of individuals. John Kunz and Kristy Kocher of Intellicorp provided support in using KEE and in knowledge-based system principles and design. Paul Marshall and Sharon Lafuse of JSC were the domain experts on the Space Station Thermal Testbed. Roger Remington and Renate Roske-Hofstrand of the Aerospace Human Factors Research Division at Ames offered insight into the graphics interface. Carla Wong, Manager of the Systems Autonomy Demonstration Project Office, and Peter Friedland, Chief of the A.I. Research and Applications Branch, gave overall guidance and support for the development of the TEXSYS prototypes.

References

[1] Bobrow, D.G., *Qualitative Reasoning About Physical Systems*, MIT Press, Cambridge, Massachusets, 1985.

[2] Bull, J.S., Brown, R.M., Friedland, P., Wong, C.M., Bates, W., Healey, K.J., Marshall, P., "NASA Systems Autonomy Demonstration Project: Development Of Space Station Automation Technology," Second AIAA/NASA/USAF Symposium On Automation, Robotics, And Advanced Computing For The National Space Program, March 9-11, Arlington, VA.

[3] Coleman, W.D., Cloyd, L.M., Chao, D.C., Ritrivi, C.A., McAdams, C.C., Patrick, J.A., "Initial Two Phase System Final Test Report," NASA Contract NAS9-17505, November 1986.

[4] Marshall, P.F., "Space-Constructible Heat Pipe Radiator Thermal Vacuum Test Program," 14th Intersociety Conference on Environmental Systems, San Diego, California, July 16-19, 1984.

[5] Rankin, J.G., "Space Station Thermal Management System Development Status and Plans," 15th Intersociety Conference on Environmental Systems, San Francisco, California, July 15-17, 1985.

[6] Starks, S.A., Rundus, D., Erickson, W.K., Healey, K.J., "NASA Systems Autonomy Demonstration Program: A Step Toward Space Station Automation," SPIE Advances In Intelligent Robotics Systems Symposium, Oct. 26-31, 1986, Cambridge, Massachusetts.

N88-24190

Integration of Symbolic and Algorithmic Hardware and Software for the Automation of Space Station Subsystems

Hugh Gregg, Lawrence Livermore National Laboratory
P.O. Box 808, L-310, Livermore, CA 94550
Kathleen Healey, Johnson Space Center
Houston, TX
Edmund Hack, Lockheed EMSC
Houston, TX
Carla Wong, Systems Autonomy Demonstration Project Office
NASA Ames Research Center
M/S 244-18, Moffett Field, CA 94035

Abstract

Traditional expert systems, such as diagnostic and training systems, interact with users only through a keyboard and screen, and are usually symbolic in nature. Expert systems that require access to data bases, complex simulations and real-time instrumentation have both symbolic as well as algorithmic computing needs. These needs could both be met using a general purpose workstation running both symbolic and algorithmic code, or separate, specialized computers networked together. The latter approach was chosen to implement TEXSYS, the thermal expert system, developed by NASA Ames Research Center in conjunction with Johnson Space Center to demonstrate the ability of an expert system to autonomously control the thermal control system of the space station. TEXSYS has been implemented on a Symbolics workstation, and will be linked to a microVAX computer that will control a thermal test bed. This paper will explore the integration options, and present several possible solutions.

Introduction

As part of NASA's Systems Autonomy Demonstration Project (SADP), a series of four demonstrations will be conducted to show the capability of increasingly complex expert systems applied to space station subsystems needs. The first of these demonstrations, the thermal expert system (TEXSYS), is being developed to show the use of artificial intelligence technology in the operation and management of the space station thermal control system. This demonstration is a joint project of the Ames Research Center (ARC) and Johnson Space Center (JSC) under the direction of the Systems Autonomy Demonstration Project Office at ARC. TEXSYS [2] will be used to monitor, control, diagnose, and reconfigure a large space station prototype thermal test bed (TTB) at JSC. A small thermal test bed will be used at ARC during the development and testing of TEXSYS prior to integration with the large TTB system at JSC.

PRECEDING PAGE BLANK NOT FILMED

The thermal expert system being developed at ARC will consist of an expert system (TEXSYS) and an intelligent human interface (HITEX). TEXSYS contains the thermal domain knowledge provided by the Crew and Thermal division of JSC, and will be used for autonomous control, diagnosis and reconfiguration. HITEX is the "human interface" to the thermal expert system, providing explanation facilities and system status graphics. This latter system will be developed by the Human Factors Division at ARC to demonstrate an ergonomic interface to the TTB, such as would be required by the thermal engineer for use on the space station. The JSC test bed will be controlled by the TEXSYS Data Acquisition System (TDAS [6]).

The thermal test bed at ARC is small version of the two-phase ammonia thermal bus prototype at JSC. The ARC TTB will be controlled by a conventional computer control system, and data collection and communications software will be developed by Lawrence Livermore National Laboratory. This paper describes the approach taken for integration of both the symbolic and algorithmic hardware and software used for the Ames thermal brassboard portion of this project. It is expected that this approach and its extensions will be compatible with and meet the requirements of the TEXSYS system when integrated into the JSC thermal test bed facility as described previously [6].

Identifying Algorithmic vs. Symbolic Processes

The three major sections of this project are TEXSYS, HITEX and the TTB control functions. All three of these processes could conceivably be executed on one computer, but overall system performance would not be adequate. It was decided that each of these functions would be handled by separate computers integrated into one larger system. This approach allows the use of optimized hardware for each subsystem, and provides the capability for the eventual integration of cooperating expert systems, such as the 1990 SADP demonstration for cooperating expert systems involving the management of both the thermal and power systems.

Separating algorithmic and symbolic processes to different computers has been shown previously to be very effective. The TQMSTUNE expert system developed at Lawrence Livermore National Laboratory [1,4-5,7-8] runs on a Xerox LISP workstation, and communicates with a conventional minicomputer (DEC PDP-11/23) that operates the triple quadrupole mass spectrometer (TQMS). The "tweaker" (a process of adjusting an instrumental parameter for maximum signal) was originally implemented, in LISP, on the Xerox workstation. When the tweaker was recoded in FORTRAN and ported to the control computer, the time required to tune the instrument (the goal of TQMSTUNE) was reduced by a factor of 6. This time difference reflected the symbolic or heuristic portions of the expert system's ability to perform better with high level information rather than large amounts of low level data.

These concepts are being applied to the TEXSYS system. For example, the control and collection of data from the thermal test bed is an algorithmic process. These processes, as well as data reduction, storage and display are well understood and are being coded in standard algorithmic languages, such as FORTRAN and C. A conventional microVAX II algorithmic processor was chosen for this application.

The heuristics of control, fault diagnosis, and reconfiguration are represented as symbolic processes. These processes comprise the heuristic part of TEXSYS and were designed using the Knowledge Engineering Environment, KEE (Intellicorp) and a NASA-developed model tool kit (MTK) for model-based reasoning on a Symbolics workstation [3]. The size, complexity, and symbolic nature of TEXSYS require that it remain on a specialized Symbolics LISP processor but must be able to freely access high level information abstracted from all the data being acquired from the microVAX.

The choice of algorithmic vs. symbolic workstation is less clear for HITEX, for it has simultaneous symbolic and algorithmic display needs. Therefore, a general purpose workstation (Sun Microsystems) able to handle both expert system development environments (i.e. KEE) and display process control graphics, will be utilized.

Communications Requirements

Each of the three systems briefly described above must be able to communicate with the others. Both the expert system TEXSYS and the human interface HITEX require current thermal test bed information, and both must be able to control the TTB. Each expert must be able to interact with the other, i.e. HITEX may be requested to explain a TEXSYS action, or HITEX (or the human operator) may need to instruct TEXSYS to change operational modes. Figure 1 shows the logical communications links required to implement this system and a few of the Ames TTB hardware

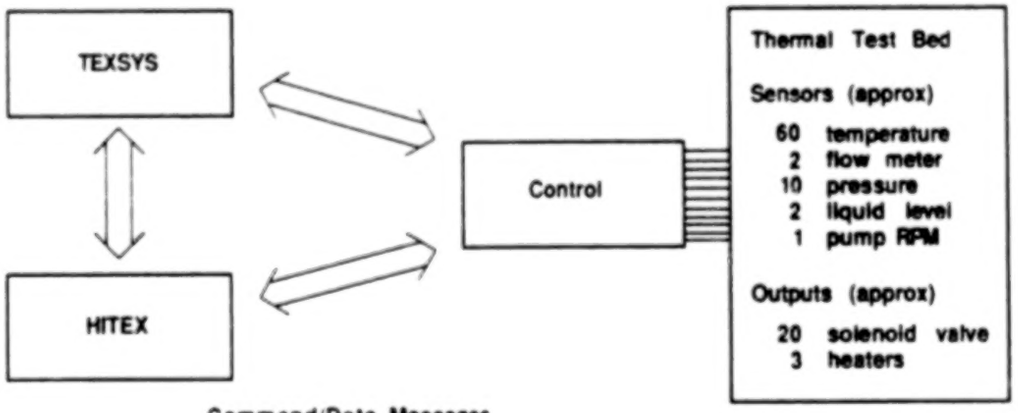


Figure 1

parameters that must be controlled. The same types of parameters must be controlled on the JSC prototype TTB with similar data rates, but there are considerably more parameters on the JSC test bed.

The control computer will be acquiring sensor input at a rate of once per second. The volume of data recorded make it impractical for TEXSYS and HITEX to evaluate the raw data for every decision. A second function of the control computer, therefore, is to extract meaningful information from the collected data. Examples of this data reduction include calculating the rate of change for a temperature sensor and checking all sensors for under or over limit conditions. With both the raw and processed data available, the expert system and the human interface can utilize whichever data is appropriate for a given rule or time constraint. A third function of the control computer is to moderate conflicting control commands from the two higher level systems.

Both TEXSYS and HITEX will need to be able to control the TTB. Control functions include opening and closing valves, turning pumps on and off, changing set points, alarms and limits and initializing the entire system. Although it is possible to have both systems control the TTB at the same time, the control software will allow only one system to issue control commands. This allows the expert system, TEXSYS, to be in control, and know the state of the TTB. If the HITEX system needs to issue a command, it must request that TEXSYS issue that command. If for some reason TEXSYS is unable to control the test bed (i.e. the computer crashed), the data acquisition computer will detect the lack of activity, and allow HITEX to control the TTB.

Communications Hardware

There are several possible hardware communications schemes, two of which are shown in Figure 2. In point-to-point topology, each process would have a dedicated link for every communication

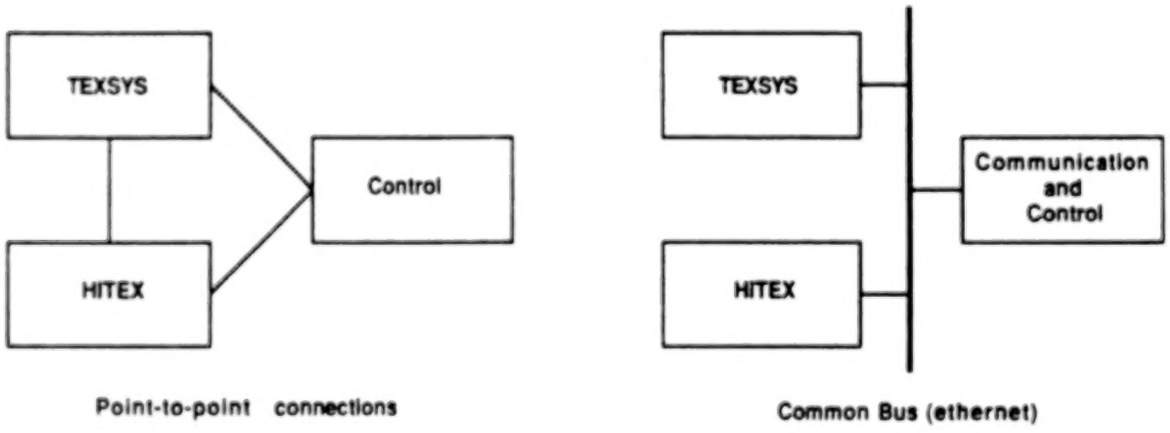


Figure 2

ORIGINAL PAGE IS
OF POOR QUALITY

channel. This is the simplest mechanism, but when the system grows beyond a few nodes, it becomes unmanageable. A bus oriented topology offers an easy growth path (for adding more systems) and often allows for logical point-to-point links. However, as a general purpose shared resource, a common bus may become a bottleneck and reduce overall system performance. Based on these considerations, a combination of a common bus and a point-to-point link was adopted for the TEXSYS project.

The choice of the communication scheme implemented is dependent on the required throughput of the communication channel. The throughput includes not only the bandwidth of the communications hardware, but also includes the software overhead associated with accessing that device. Each of the processors selected for this project are able to use ethernet as a communications bus. The protocol used to send packets over the ethernet will be either TCP/IP or DECNET, depending on the throughput of each package.

TEXSYS may require large quantities of information from the data acquisition/control computer. A specialized point-to-point link between those two computers will be used. This "bus-link" (made by Flavors Technology, Inc.) allows the Symbolics computer direct access to a portion of the microVAX memory. By copying the TTB parameters into a specified portion of the microVAX memory, TEXSYS is able to quickly retrieve any required datum. Control commands and communications with HITEX will use ethernet to provide consistent interface among all three systems. The use of both ethernet and the bus-link product give TEXSYS the best of both systems: a point-to-point link for accessing large quantities of data, and a common bus to provide an interface to other systems.

Communications Software

Expert systems running under the KEE shell are able to take advantage of a KEE feature, active values. When a rule (or LISP code) either gets or puts the value of a slot (e.g. reads or writes the value of a variable), the active value method (e.g. subroutine or function) is invoked if it exists. This active value method may execute any code, and the the value it returns (for a GET.VALUE function of a slot) is used by the rule that requested the value. Both TEXSYS and HITEX use this mechanism to retrieve TTB parameters. For example, if a TEXSYS rule premise is based on the pressure before the condenser (eq. 1), the rule interpreter retrieves the value of the transducer (eq. 2) and the active value methods fires (eq. 3). This LISP code either looks up the value in the shared memory provided by the Bus-Link, or uses ethernet to request the value from the control computer. The active value method then returns a value, P, to the slot (eq. 4), and the rule uses the returned value for its comparison (eq. 5). Values may be set in the same way.

```
if Pressure_before_condenser > limit then conclusion    (1)
GET.VALUE Pressure_before_condenser                     (2)
AVGET method (LISP code, returns P)                     (3)
Pressure_before_condenser = P                           (4)
if P > limit then conclusion                            (5)
```

The use of active values allows for transparent use of the network to obtain the needed information, and can be used for all routine messages and data between the human interface and expert system and the control computer. This method was used for the TQMSTUNE system described earlier. It should be noted that the active value mechanism imposes a master/slave relationship upon the systems.

Both the human interface and the expert system act as a communication master (requestor), sending or receiving data as necessary to derive explanations of to test the rules. The control computer, however, must be able to quickly respond, at any time, to requests from one or more external systems. As a slave process (data server), the control computer cannot send any data unless requested by the master process. In the event of an alarm condition (i.e. sensor out of limit), the control computer must wait until the expert system requests some data, and then may send a warning flag indicating a potential problem. The expert system may then request more information about the warning. Since the expert system is constantly requesting information and data from the control computer, the warning condition will be recognized in short order. This configuration assumes that the delay in the notification of a warning condition is minimal when compared to the overall response time required to service the alarm condition by the expert system.

A second method of communicating between the expert system and the control computer is to have the expert system explicitly request information from the communications interface. This allows the expert system complete control over the knowledge base, and all information in it will be consistent. A separate process can receive data and alarms from the control computer, and the diagnostic portion of the system initiated if an alarm occurs.

Communications Subsystem

The active value mechanism works well for communications between the expert system and the control computer, but for lengthy, information messages (i.e. TEXSYS explaining to HITEX why certain actions were taken), a different approach should be taken. Two possible solutions are a direct logical connection between TEXSYS and HITEX over ethernet, or a microVAX buffered communications system between the human interface and the expert system. Both configurations are under consideration.

Ethernet is used as the common communications bus; a communications interface, implemented on the microVAX, is used as

the software communications bus. Figure 3 is a block diagram of this interface software. The communications interface acts as a central clearing agent for informational messages, warning messages and commands. Information messages, from whatever source, are saved until requested, while warning messages and commands are immediately sent to the target processes. In this way, only urgent messages (warnings and commands) interrupt the expert system, and routine informational messages are retrieved only when the expert system needs that information.

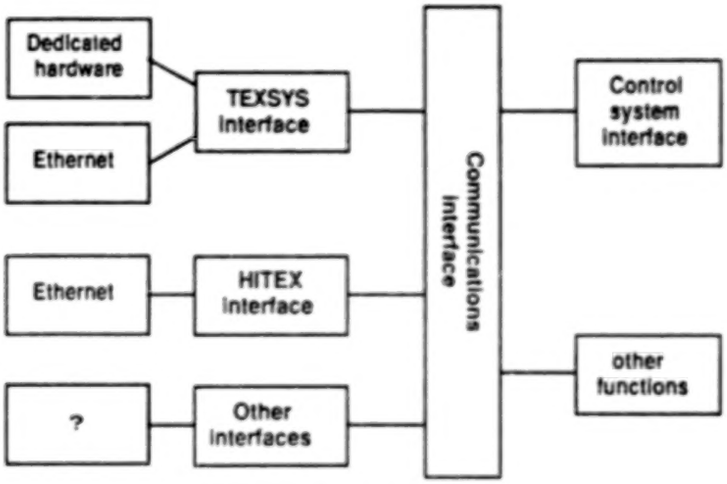


Figure 3

All systems interfaced to this message router have access to the TTB parameters and status (via the control system interface). Daemon processes analyze the collected data ("other functions" in Figure 3) and issue warning messages when needed (i.e. if a pressure is rising too rapidly or is out of acceptable limits). In theory, commands to control the TTB may be given by any connected process, but command conflicts would be a major problem, and would best be handled by the expert system. For this reason, it is expected that all commands will originate from TEXSYS, and HITEX will request TEXSYS to issue commands as necessary.

The dedicated TEXSYS to control computer link is essentially a virtual memory device. This link is in addition to the ethernet link, and bypasses much of the software overhead associated with the ethernet link. TEXSYS could operate without this link, but its use allows a significant decrease in TEXSYS access time to TTB parameters.

Conclusions

The division of processes among algorithmic and symbolic processors is usually straight forward. Accessing data bases or real-time instrumentation are algorithmic processes, while expert systems are generally symbolic in nature. To effectively integrate these these processes into a composite system requires an effective communications scheme. For small systems, point-to-

point links are simple and sufficient; however, for large systems, bus topology and a message server are usually required.

Acknowledgements

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-ENG-48. The support of the the Systems Automony Demonstration Project Office at NASA Ames Research Center (proposal L-792) is gratefully acknowledged.

References

[1] Brand, H.R. and Wong, C.M., "Application of Knowledge based Systems Technology to Triple Quadrupole Mass Spectrometry", Proceedings of American Association of Artificial Intelligence (AAAI-86), Vol. 2 (1986) 812-819.

[2] Bull, J.S., Brown, R., Friedland, P., Wong, C.M., Bates, W., Healey, K.J. and Marshall, P., "NASA System Autonomy Demonstration Project: Development of Space Station Automation Technology", 2nd AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program; Arlington, VA, March 9-11, 1987.

[3] Erickson, W.K. and Schwartz, M.R., "MTK: An AI Tool For Model-Based Reasoning", 3rd Annual Conference On Artificial Intelligence for Space Applications, NASA/MSFC, Huntsville, AL, Nov. 2-3, 1987.

[4] Gregg, H.R., Brand, H.R. and Wong, C.M., "A Knowledge based System for Tuning MS/MS Instruments in Various Operational Modes", 34th Annual Conference on Mass Spectrometry and Allied Topics, Cincinnati, OH, June 8-13, 1986.

[5] Gregg, H.R. and Wong, C.M., "TQMSTUNE: The Rules", 35th Annual Conference on Mass Spectrometry and Allied Topics, Denver, CO, May 24-29, 1987.

[6] Healey, K. and Hack, E., "TDAS: A Thermal Expert System Data Acquisition System", SOAR-87: 1st Annual Workshop on Space Operation, Automation and Robotics; NASA/USAF/Univ Houston, Houston, TX, August 5-7, 1987.

[7] Wong, C.M., Crawford, R.W., Kunz, J.C. and Kehler, T.P., "Applications of Artificial Intelligence to Triple Quadrupole Mass Spectrometry", IEEE Transactions on Nuclear Science, Vol NS-31, 1, (1984), 804-810.

[8] Wong, C.M., Lanning, S.M., Crawford, R.W. and Brand, H.R., "Application of Artificial Intelligence Programming Techniques to the Development of an Expert System for Tuning a Triple Quadrupole Mass Spectrometer", 32nd Annual Conference on Mass Spectrometry and Allied Topics, May 1984.

N88-24191

Connecting Remote Systems for Demonstration of Automation Technologies

R. M. Brown and R. Yee
Systems Autonomy Demonstration Project Office
NASA/Ames Research Center
Moffett Field, CA

ABSTRACT

Work will begin this year on the development of the second of four demonstrations of automation technology under the Systems Autonomy Demonstration Project (SADP). This demonstration will involve elements of four NASA Centers: ARC, JSC, LeRC, and MSFC. Intercenter digital data communications will be a vital element of this demo.

This paper presents an initial estimate of the communications requirements of the SADP development and demonstration environments, a proposed network paradigm is developed, and options for network topologies are explored.

INTRODUCTION

The Systems Autonomy Demonstration Project (SADP) was established to conduct a series of demonstrations of the use of advanced automation technology in solving problems applicable to the Space Station. The first demonstration, scheduled for 1988, will use expert system technology and model-based reasoning to monitor and control the operation of Johnson Space Center's Thermal Test Bed.

Work will begin this year on the second SADP demonstration, this one scheduled for completion in 1990. Unlike the 1988 demonstration, intercenter digital data communications will be a vital element of the 1990 demo. To accomplish the 1990 demo, elements of Lewis Research Center (LeRC) and Marshall Space Flight Center (MSFC) will be included in the SADP. The demonstration itself requires the interaction of systems located at LeRC and JSC, and cannot be accomplished without intercenter data communications.

Within NASA, intercenter data communications are provided through the Program Support Communications Network (PSCN). The PSCN employs terrestrial and satellite transmission facilities to support all elements of the agency and provides a wide variety of services, including intercenter telephone, FAX, voice and video teleconferencing, electronic mail, and digital data communications. The PSCN is based on a foundation of equipment and leased lines tying together all sixteen major NASA locations.

SADPNET REQUIREMENTS

In designing a network like SADPnet, the first step is to define the overall network goals, including functions, connectivity, interfaces, operational quality and cost, expansion capability, and implementation cost.

Hardware capabilities

To understand the limits posed by the systems at each site, a study was made of the file transfer capabilities between a Symbolics 3600 system and two other computers on a 10-Mbit/sec Ethernet LAN. These measurements were made using the TCP-IP protocol and FTP service. The results are shown in Table 1.

Table 1 -- Measured file transfer rates

Path	No of trials	File size (bytes)	Av time (sec)	variance (sec)	transfer rate (bytes/sec)
1 -> 2	7	0	0.780	0.003	n/a
1 <- 2	6	0	2.345	0.738	n/a
1 -> 2	10	1210	1.165	0.010	1039
1 -> 2	5	1209 KB	214.1	10.2	5647
1 -> 3	1	1209 KB	111.0	0.0	10892

System 1 is a Symbolics 3600 running Genera 7.1
System 2 is a DEC microVAX II running Ultrix 2.0-12
System 3 is a DEC VAX 11/780 running VMS 4.5

It is clear that the measured transfer rates are relatively low, reflecting in part the overhead posed by the Symbolics operating system. However, the transfer rate between the Symbolics and the VAX for a large file was double that to the microVAX for the same file, showing that the microVAX also limited the transfer rate. These data are preliminary, and tuning of the systems may provide improvements. However, they suggest that the overall performance of the SADPnet may be strongly constrained by the computers at each end of the link.

End-user functions

For SADPnet, three major end-user functions dominate the design: process-to-process communications services; virtual terminal services; and file transfer services.

The links that will be needed between expert systems are examples of process to process (p-p) communications. Figure 1 shows one possible requirement for links between JSC and LeRC to

connect testbeds at each site. When connected, the expert system controllers will share information and coordinate actions in a manner similar to that needed in a Space Station environment.

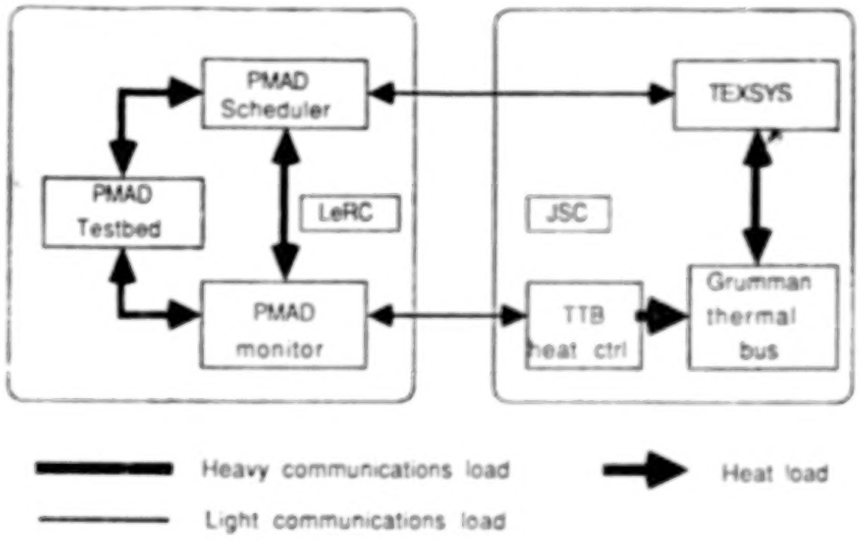


Figure 1. Possible 90 Demo p-p communications

Because these systems have little direct interaction, there is no apparent need for high data rate between the TEXSYS controller and the PMAD controller. After initial setup, the average data rate will probably not exceed one packet per second. Link periods will range from two hours, during the installation and integration activities, up to several days for the demonstrations. Outages during this period must be avoided or the demonstration activity may have to be aborted and rescheduled.

In addition to the general p-p links, SADP needs to have a virtual terminal (VT) service. VT service allows any of a wide range of terminals at one site to act as if it were connected to equipment at another site. A VT session will need to provide an equivalent to 9600 baud service with link periods of a few minutes to a few hours.

Because it is so visible to the end user, VT service will be difficult to provide. Outages, failures to connect, or other communications failures will interfere directly with the efficiency of project staff. In addition, personnel who use this service on a local area network get close to 100% availability and reliability, and will likely use this as the criterion for success when evaluating an SADPnet implementation.

The third service required for SADPnet is a reliable file transmission capability. Large file transfers will occur infrequently. The distribution of major system builds, for example, are expected to occur no more than once per month, on the average. These files are expected to be approximately 2-10 MBytes in size, and data transfer times for these large files should be accomplished in no more than a few hours.

Small files will be transferred more often. These files, less than 1 MByte in size, should be transmitted in a few minutes or tens of minutes. The transfer rate implied by the above requirements is modest, less than 15 Kbits/sec.

It is reasonable to expect file transfers to be efficient users of the provided bandwidth of SADPnet. For example, large files were recently sent between Langley Research Center and Ames Research Center over a dedicated 224 Kb/s PSCN line using TCP-IP protocols. The measured transfer rate was 219 Kb/s, exactly as predicted by the percentage of overhead in the packet.

Connectivity and Interfaces to Intra-center networks

Each center involved in SADPnet will need to communicate with at least 2 other centers via SADPnet. In fact, the only path where a need for connectivity has not yet been established is the MSFC-JSC link.

A possible configuration of the network at JSC for the 88 demonstration provides an example of the local elements of the overall SADP system. Figure 2 shows a simplified diagram of the elements of the TEXSYS system, together with other elements of the Thermal Test Bed. The configuration at ARC is similar -- an Ethernet bus with computers and controllers directly attached.

Simultaneous service

The SADPnet cannot be effectively used if it is the equivalent to a large party-line where only one connection can be supported at any moment. The number of simultaneous connections to be maintained by SADPnet for the 90 demo will be determined by further studies, but is expected to be less than ten.

Protocols

The SADP communication service will have to support several protocols. Digital Equipment Corp.'s Digital Network Architecture (DNA) and the ARPAnet TCP/IP protocols are currently used. As the SADP progresses, it may be found to be appropriate to convert to the ISO standard. Therefore, it is prudent to design a network that will simultaneously support all three. This is not an unreasonable requirement, as both TCP-IP and DNA are designed to be co-resident with multiple protocols.

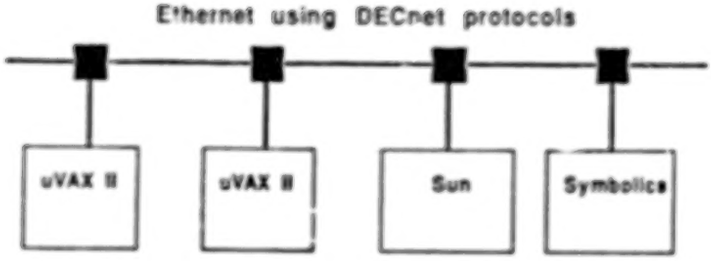


Figure 2 -- Possible network configuration for 88 demo

Reliability, Availability, and Maintainability

A basic requirement for the SADPnet is that it provide reliable service, that it be available when requested, and that an maintenance organization is available and competent to repair the service when it fails. These requirements are typically known as Reliability/Availability/Maintainability, or RAM.

Error rate is generally the metric used when considering reliability. Based upon recent studies of seventeen PSCN links, a reasonable expectation for packet error rate is that it will not exceed one packet in a thousand. The PSCN goal for packet error rate -- based upon ATT standards -- is that it not exceed 0.5 packets per hundred.

Availability is the probability that the service will be available for use when needed, and that no outages (as opposed to burst errors) will be encountered once the connection is established. An appropriate availability goal is that the system be available for use at least as often as telephonic access to the same location. This implies that a request for connection to another site should be satisfied with an 0.99 probability at an 0.50 confidence level.

Finally, the maintainability of the service should approach that provided by most long distance digital common carriers. Though the PSCN is a new network, the service provided by PSCN operations staff is approaching the level needed.

Expandability

Expansion of the SADPnet is a distinct possibility. It therefore should be designed now to allow this expansion -- adding sites, services or levels of performance. Expansion should be possible without hurting the existing service and without unusual cost impact.

PSCN RELIABILITY

Accurate data are available to judge the PSCN performance in terms of error rate. These come from an unpublished study[1] of NASnet, a communications network using PSCN that ties computers at ARC to 17 other sites, including all of the sites associated with SADP. These links include dedicated T1 services, switched 56 Kb/s services staying on the PSCN backbone, switched 56 Kb/s service with tail circuits from the backbone, and dedicated 224 Kb/s lines through the PSCN backbone. All of these links are provided through terrestrial, rather than satellite, services.

Table 2, below, summarizes the NASnet measurements of the PSCN error-rate on 15 of the 17 circuits over a 30 day period. Two of the 17 sites had no traffic during this period.

Table 2 - NASnet traffic statistics

Site	up time	MBytes sent	MBytes rcvd	% CRC errors	Circuit type
1	736.0	253.3	155.0	0.00	Ded. ATT T1
2	720.5	5740.2	1272.4	0.02	Ded. 224 Kb no tail
3	699.5	50.2	10.5	0.92	Sw. 56 Kb
4	517.9	1965.0	83.2	0.18	Ded. 224 Kb no tail
5	445.5	14.2	15.2	0.00	Sw. 56 Kb no tail
6	226.0	30.1	18.0	0.00	Sw. 56 Kb
7	90.3	32.8	10.5	0.01	Sw. 56 Kb
8	82.8	21.1	6.8	0.02	Sw. 56 Kb
9	58.7	38.2	13.8	0.01	Sw. 56 Kb
10	44.3	74.1	19.3	0.01	Sw. 56 Kb
11	41.5	13.3	5.7	0.05	Sw. 56 Kb
12	37.8	69.3	56.2	0.50	Sw. 56 Kb
13	34.6	42.6	44.2	0.01	Sw. 56 Kb
14	16.7	12.2	40.0	0.02	Sw. 56 Kb
15	4.3	0.0	0.0	0.00	Sw. 56 Kb no tail
16	2.8	0.2	0.1	0.58	Sw. 56 Kb no tail
17	0.0	0.0	0.0	0.00	Sw. 56 Kb

One conclusion that can be drawn from this study is that the dedicated and switched 56 Kb/s PSCN lines can provide error rates that are acceptably low when measured against SADPnet requirements. Eleven of the fifteen sites demonstrated packet error rates less than one in a thousand, and all had error rates less than one in a hundred. The PSCN goal for error rate is 1 in 200, limited by the commercial carrier offering.

---

[1] Data here were provided by Judy McWilliams, General Electric Corp, from studies covering the 15 week period from May 24, 1987 through Sep 4, 1987.

However, the study reveals some problems in terms of availability. The three dedicated line circuits in this network should have had 100% availability; only one of these lines achieved that goal.

That line (to Site 1) provides a standard by which the others can be judged. It is a dedicated T1 service running over ATT lines that had an uptime of 100%. No packet errors were seen on this line under moderate to heavy system loads.

On the other hand, site 2 had over 15 hours of downtime and a resultant availability of 98%, while site 4 achieved only 70% availability. The level of service was reported to be considered 'good' by the users at site 2 and 'poor' at site 4.

SADPNET OPTIONS

Since the switched and dedicated line services of PSCN can provide connectivity with acceptably low error rates, the design process now involves establishing a network paradigm, topology, and channel speed; then fleshing these out with hardware and software options.

SADPnet Paradigm

Because the goal of SADPnet is to connect LANs at each site, the basic network paradigm proposed is that it be an extended Ethernet service, though one that has a lower bandwidth for inter-center traffic than that provided intra-center. This paradigm permits the use of existing hardware, software, and protocols, and allows for the addition or removal of computer equipment at each site without coordination problems. In addition, experience with this approach has shown it to be both feasible and economical.

This paradigm involves three major elements, as shown in Figure 3. The top element is the communications network provided by PSCN lines, whether dedicated or switched. The bottom element is the local network, containing the existing and planned development and testbed computers. In between is an Ethernet bridge device and communications interface.

The bridge is the critical element in this network. It transmits to the other sites only those packets being sent to remote systems. This keeps local Ethernet traffic from being transmitted through the network, a feature that reduces network bandwidth requirements enormously. The bridge makes use of configuration information to manage the network traffic, and to provide network statistics and security.

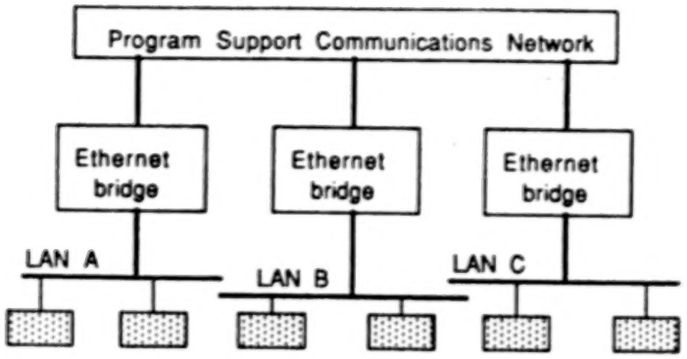


Figure 3 -- Basic SADPnet paradigm

Possible Topologies

The network topology can be a star, a bus, or a series of point-to-point links. Figure 4 shows four possibilities.

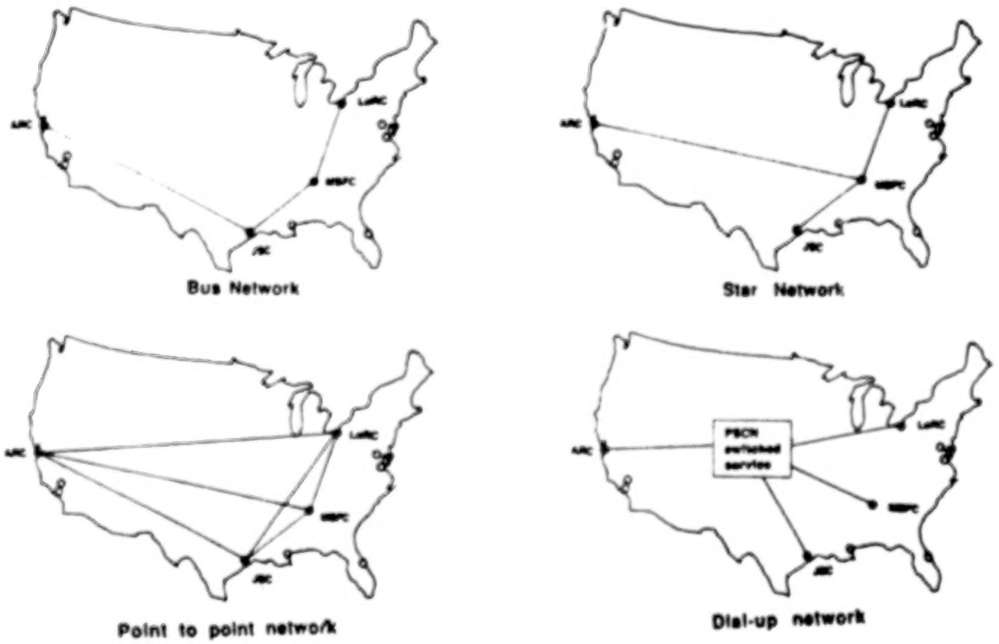


Figure 4 - SADP Topology Options

The bus, star, and point-to-point options shown in Figure 4 can meet all SADPnet requirements; the dial-up option can meet them under conditions of very light load. Choosing among these options will require cost/performance tradeoffs, and a complete analysis of these tradeoffs is beyond the scope of this paper.

CONCLUSIONS

This study has focussed on communications requirements for the SADP 1990 demonstration. The next step is to validate the requirements, explore the network options, and select a design. A project team, including members from each site, should then be formed to implement the network, a schedule and budget for this project established, and implementation begun.

SADPnet will be the first within NASA to be used to connect interacting automated controllers, and to do so over long distances. The SADPnet has every chance of meeting technical requirements, cost constraints, and schedule requirements, if the steps noted above are initiated promptly.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the help provided by the NASA/ARC NAS Systems Division by making available the unpublished results of the tests of PSCN reliability and transfer rates.

N88-24192

Knowledge Based System Verification and Validation as Related to Automation of Space Station Subsystems: Rationale for a Knowledge Based System Lifecycle

Keith Richardson
Carla Wong

Systems Autonomy Demonstration Project Office
NASA/Ames Research Center
Moffett Field, CA 94035

Abstract

The role of Verification and Validation (V&V) in software has been to support and strengthen the software lifecycle and to ensure that the resultant code meets the standards of the requirements documents. Knowledge Based System (KBS) V&V should serve the same role, but the KBS lifecycle is ill-defined. The rationale of a simple form of the KBS lifecycle is presented, including accommodation to certain critical KBS differences from traditional software development.

Introduction

This paper discusses the rationale for using a KBS lifecycle to solve some problems of KBS V&V, describes the agents of this process, and presents three key aspects of variance from traditional software lifecycles. The KBS lifecycle is described in greater detail in a longer paper titled : "Workshop on Verification and Validation of Knowledge Based Systems: Recommendations for Procedures and Research."

V&V is the process of overseeing the construction and testing of a software program according to specifications. The purpose of V&V is to ensure that software is:

1) Designed to be testable,
2) Completed according to defined specifications which lead to testability,
3) Tested according to requirements.

The software lifecycle is used to structure this process. Since such a lifecycle has not been generally accepted for KBS it was natural that one of the major topics of conversation during the Ames KBS V&V workshop was definition of a KBS lifecycle. (Two proposals which outline some aspects of KBS lifecycle are discussed in the workshop summary.)

The KBS lifecycle presented here, however, addresses V&V issues in the context of a complete, high-level, lifecycle description. This lifecycle may be seen as the simplest form, applicable to a single KBS acting without interaction with other systems, but including the human interface. Some other complex situations will only need reasonably straightforward adaptations of the lifecycle, while others using new AI methodologies or with

PRECEDING PAGE BLANK NOT FILMED

PAGE 24 INTENTIONALLY BLANK

interdependencies to external systems will not be accommodated by a KBS lifecycle without the benefit of further experience and research.

For the purposes of this paper the lifecycle is further simplified to the following structure:

- Requirements Phase
- Prototype Phase
- KBS Build Phase
- Test Phase
- Delivery and Monitor Phase

The basic format of the KBS lifecycle is derived from traditional software lifecycles, both for the reason that these lifecycles are better understood by the sponsoring organizations and software developers and because there are many aspects of KBS software methodology which are similar, or the same as, traditional software. Where software development methods are similar it is cost effective to capitalize on this extensive experience in traditional software.

Satisfying Agents of KBS Development

Among the critical problems which must be solved by a KBS lifecycle is the satisfaction of all the agents in the process. The major three agents are:

1) The agents who sponsor the project: the organization, managers, and investors.
2) The agents who build the KBS, the domain experts, knowledge engineers, test engineers, etc.
3) The end users of the KBS.

A person may serve as more than one agent in the KBS development.

The reason for satisfaction of the first set of agents is self-evident.

Satisfaction of the KBS builders is necessary because there are some aspects of the KBS development only they are familiar with - not every programming decision can be recorded. Some have suggested that independent testing and V&V agents will have a better understanding of the KBS than the builders; what they mean is that certain blind spots in the developer's perception are better discovered by people with different blind spots. Among the ultimate authorities for the integrity of a KBS must be the people who have been involved in the experimental process of prototyping - no other group will understand design decisions in such precise detail. For this reason the KBS lifecycle should support testability as defined by the KBS requirements, and support only secondarily independent code evaluators.

The third set of agents is the users. User acceptance is the final step in the KBS process before the KBS will be used and can be called successful. The importance of the users to the acceptance of a KBS can be more important than in acceptance of traditional software in three related ways:

1) Some users, e.g. astronauts, mission controllers, or doctors and lawyers, (and their professional organizations, the AMA, and ABA) have almost complete authority to decide whether software will be used in their domain - independent of their ability to substantiate their concerns.

2) A KBS may need to perform a range of functions, such as diagnose, monitor, control, educate, etc. Which of these functions are important, and in what manner they may be fulfilled is largely the decision of the users.

3) Users must evaluate how effectively they are able to use the KBS information in their real world domain.

The satisfaction of these three sets of agents, the sponsors, the builders, and the users, is the goal of the KBS lifecycle. This satisfaction should provide direction to all phases of the KBS lifecycle, and each stage must contribute to the end result of the process.

Three Aspects of the KBS Lifecycle

Among the important aspects of adaptation of traditional lifecycles to KBS development are:

I) Changes and improvements to requirements documents.
II) Addition of a reiterative prototyping loop.
III) Inclusion of the user in the development process.

I) Requirements Documents.

Some in the Ames workshop felt that the documents which are produced before prototyping in the KBS development process needed to be more completely specified - some went as far as to say that inadequate requirements were the primary cause of the failure of KBS development efforts. Others insistently pointed out that the result of the experimental process cannot be predicted before prototyping, and that precise specifications were unlikely or impossible.

Issues can be confused by this simple polemic distinction between requiring more or less thorough requirements documents. The suggestion made by this paper is that a different sort of requirements document is necessary which at once suits the experimental nature of KBS development, the pragmatics of funding and time constraints, and the necessity to be able to test function and assess the KBS success.

Experimental concerns dictate that the KBS development process be as unconstrained as possible and able to exploit fortuitous discoveries during the prototyping phase. Because neither the domain experts or the knowledge engineers have a precise idea of which experimental areas will be practicable at the beginning of the prototyping phase, enough time and resources must be available to pursue the certain number of investigations which will not contribute directly to the final KBS. Any requirements documents should be written to accommodate the prototyping process, and should specify money, time, and other resources available for prototyping.

Constraints dictated by KBS development sponsors, besides including restrictions on resources, should also specify as closely as possible criteria for success. In a "proof-of-concept" KBS development project , such as the Systems Autonomy Demonstration Project, the criteria for success will be the proof or dis-proof of the practicality of KBS application in an environment generally - no specific function may be required of the final stage of the KBS. A commercial KBS project intended for delivery and/or sale will be measured by utility functions such as: customer satisfaction, number of units sold, profit, repeat sales, maintenance costs, etc. KBS intended for NASA will use various criteria of success which will depend on program goals, intended use, hazard analysis, etc.

The requirements documents, where it measures KBS success in terms of utility, must take in account the uncertain process of the prototyping phase. It should include statements of goals as well as of resources. The format which is liable to be crucial to KBS development is the specification of a range of results, or a number of solutions. The sponsors of KBS development need to put considerable thought into the various combinations of constraint satisfaction which will prove satisfactory to them.

II) Reiterative Prototype Loop

Incremental refinement in prototyping is an inseparable feature of KBS development. Traditional software management is uncomfortable with this seemingly open-ended process, but costs and other resources can be controlled by astute specification of requirements.

The prototyping loop has also caused controversy in that the documentation style of a traditional software prototype seems to be overly cumbersome for KBS development. Traditionalists are uncomfortable with less than "complete" documentation, while KBS developers point out that design decisions early in the prototyping process which are later superceded are expensive to document and do not serve in a direct way to satisfy requirements. A goal should be that complete documentation be kept of any aspect of the project likely to be incorporated into a final version of the code. An experimental notebook may be an example of a format which satisfies all parties; decisions on what to include will be made partly by the prototype development team, and partly by system requirements.

The KBS prototype loop's duration and experimental extent will be limited by the requirements specification.

Where the requirements documents are intentionally incomplete one of the goals of the prototyping phase is to add further details to them. Thus a complete loop in the prototyping phase involves:

1) Knowledge Acquisition/Extraction.
2) Building of a Prototype.
3) Evaluation of Results.
4) Augmentation of Requirements Documents.
5) Decision on Direction of Next Prototype.

Decisions about project directions are made on the basis of newly revised requirements documents, and experimental results from the most recent prototype, and are used to determine whether: a) another prototype is needed, b) the project should be discontinued, being unable to satisfy requirements, c) the project should continue into the next lifecycle phase, which is to build a less experimentally oriented KBS which will serve as a platform for associated development efforts in interfaces, testing plans, user facilities, formal documentation, etc.

Following prototyping, changes to fundamental aspects of the KBS become less practical. During the prototyping, efforts to KBS development are restricted so that they rely on mostly inflexible assumptions, after prototyping, KBS-related efforts rely on the stability of the prototype itself. Major design decisions should be made during prototyping phase, later changes are possible, but at greater cost for associated efforts.

III) User in the Loop.

Another major difference between traditional software lifecycles and KBS lifecycles is the increased involvement of the user. It has been suggested that the end user of a KBS may have much discretionary power in approving use of a KBS. This hurdle can be anticipated and surmounted by including users in the development process.

In traditional software lifecycles precise user requirements can be included, or tacitly implied, in the requirements documents; but these documents can remain incomplete in the KBS lifecycle until a fairly late stage. At KBS requirements specification the incomplete user input is mitigated by the presence of the domain expert during prototyping, who in some ways will typify the concerns of the general user.

In traditional software another point of user input in the development process is the Alpha or Beta test, where a nearly finished version of the software is placed in an environment typical of expected use. In a KBS it is much more difficult to define "typical environment" and "expected user". One solution, indeed, is to limit KBS use to narrowly defined situations. An approach which is more appropriate for a KBS to be used in a variety of situations, and by users of varying skills, is to attempt to accommodate as many environments as possible. These accommodations will start by a direct user contribution to the requirements documents.

To the extent a KBS is liable to be used in situations which were unanticipated by the designers, or which change over the lifespan of KBS use, it is necessary to extend the KBS lifecycle to include user reaction and comment after KBS delivery. The overhead incurred by this additional burden should be designed to be of less cost than the combined benefit: to current users, to addition of new areas of utility, and of the value of direct user input for later versions of the KBS.

User interaction is liable to be least effective in the prototyping phase, where the state of the development process is complicated to explain , and the state of domain knowledge is non-coherent. The writing of requirements documents, however, provides an opportunity in that at that early point it is efficient to catch misconceptions about expectations of the user community, and relatively easy to explain project plans. The user will also have an important perspective on utility tradeoffs among various possible development options.

Conclusion

These changes, and others which can be found in our related papers, do not represent a complete plan for a KBS lifecycle. The KBS lifecycle will be improved with experience following these fundamentally important changes, especially in areas where there is not an analogous process in traditional software development. The KBS development process also will benefit from a significant amount of research over the next few years to improve predictability, costing, enhanceability, and so on. Research suggestions include improvements to supporting hardware and software, new KBS tools, automated testing facilities, and continued improvement to the KBS lifecycle.

Good V&V of KBS will follow improved specification and control of the KBS development process. Since the KBS V&V process is now being formalized it will be in the position to accommodate the latest software concerns; if the KBS V&V process seems difficult it is partly because more stringent demands are now being made on all software development. KBS V&V, then, in some ways will be the first to incorporate new standards of quality for software.

Acknowledgements

The technical assistance and comments of Dr. H. Lum, Dr. P. Friedland , and M. Dutton were much appreciated in the production of this paper.

N88-24193

TALOS: A Distributed Architecture for Intelligent Monitoring and Anomaly Diagnosis of the Hubble Space Telescope

Bryant G. Cruse
Lockheed Missiles and Space Company
Code 400.8, NASA/GSFC
Greenbelt, Maryland 20771
(301) 286-3105

ABSTRACT

Lockheed, the Hubble Space Telescope Mission Operations Contractor, is currently engaged in a project to develop a distributed architecture of communicating expert systems to support vehicle operations. This architecture, named TALOS (Telemetry Analysis Logic for Operating Spacecraft) has the potential for wide applicability in spacecraft operations. The architecture mirrors the organization of the human experts within an operations control center.

The Hubble Space Telescope (HST) is presently scheduled for launch in June, 1989. When launched it will be the most complex spacecraft yet operated from Goddard Space Flight Center. Lockheed Missiles and Space Company is the prime integration contractor for the HST and is also the Mission Operations Contractor (MOC) for Goddard.

Operating the vehicle will be a knowledge intensive task. MOC personnel will be checking more than 4900 individual telemetry parameters on 200 CRT displays against a daily Mission Timeline printout several inches thick to verify spacecraft operation. Monitoring operations will continue around the clock on an almost continuous basis.

Off-line operations consist of diagnosing malfunctions and anomalies recognized by the on-line personnel, devising recovery procedures to restore normal operation once an anomaly has been understood and to track long term trends in vehicle performance. The systems engineer must be prepared to wade through a sea of data including telemetry, on-board computer memory dumps, schedule information, orbital data, and design specifications.

The MOC began to evaluate the potential of Knowledge-based software late in 1984. Interest was anything but academic. The MOC is not a research organization. If AI could help it operate the Space Telescope with greater safety and efficiency it would be used. Otherwise, effort would continue to be concentrated on conventional approaches.

The MOC's first expert system based application for the space telescope was demonstrated in the fall of 1986. The system extracted 70 different engineering parameters from a history tape recording of the vehicle engineering downlink and analyzed them with respect to vehicle safemode events. The system contained 230 rules and was able to perform in 5 or 10 minutes an analysis that would have taken a trained engineer about an hour.

This application married a telemetry extraction program written in FORTRAN with an expert system written using LES, the Lockheed Expert System. LES was a good choice for a number of reasons. First, it runs on the HST VAXes sparing the expense and difficulty of buying and integrating a LISP machine into the ground system. More importantly, LES's knowledge representation syntax is relatively straight-forward and approachable by the average spacecraft engineer. Learning to use LES is no more difficult than learning the PSTOL operating language of the HST ground system. This characteristic saves the expense and difficulty of finding or becoming LISP programmers. Finally, "customer support" for LES from the Lockheed AI center is excellent. Numerous changes have been incorporated into LES directly to support the MOC's needs.

Encouraged by the success of the first application, the MOC has, in partnership with the ST program in Sunnyvale and

the AI Center, expanded the architecture to one which has potential capabilities for automation in all our monitoring and diagnostic functions. Work has been conducted within a number of practical constraints. Development and test of TALOS cannot be allowed to place any risk or processing burdens on the existing ground system. It must be hardware and software compatible with the existing ground system. Also to be useful in the area of real-time monitoring in the STOCC it must perform very fast if it is to keep up with the incoming data. At the same time in the off-line diagnostic mode it must be flexible and offer a good explanation facility.

While LES is well suited for the off-line diagnostic role it does not have the speed to handle the quantity and rate of the HST real-time engineering downlinks. Fortunately, a symbolic processing tool several orders of magnitude faster than the current generation of expert system shells is being developed at the Lockheed AI Center. L*STAR, with it's high speed and high resolution color graphics, is an ideal tool for the on-line portions of the architecture. Like LES, L*STAR is VAX/VMS compatible and has a user-friendly knowledge representation syntax.

If there is a "classic" approach to developing an expert system it is something like this: A knowledge engineer, who knows everything about an expert system shell but nothing (initially) about the problem to be solved, "extracts" knowledge from a domain expert, who knows nothing about AI, and captures the domain knowledge in the syntax of the shell. This approach has worked well enough for small applications of 100 rules or so but it is doubtful whether it will work on a very large system. A knowledge engineer cannot build a working system if the knowledge that goes into it is unintelligible to him. He must come to understand the knowledge of the domain expert to a certain critical level to build a truly useful system. This is not a terribly tall order for a simple machine like a soup sterilizer but for the Space Telescope it is daunting.

Providing a significant level of automation in the operation of a machine as complex as the Space Telescope could easily require 10,000 rules. The expertise that would be represented in the rules lies not in a single expert but

in tens, even hundreds, of experts throughout the Space Telescope community. These experts have years of spacecraft engineering and operational experience. A further complication is that the expertise that does exist is of a particular kind. At this point experts in operating the space telescope are really experts on how it is planned to operate and how it is expected to work on orbit. Only by sifting through experience accumulated through tests and with other spacecraft can the expert come up with a sound rule for application to the HST.

It is expected to be easier to train spacecraft engineers (who, after all, are no strangers to computers and programming) how to effectively use expert system shells than to teach professional knowledge engineers how to fly the space telescope. Experience thus far has borne out this approach.

That the TALOS architecture must be a distributed one a given due to the number and over-all complexity of desired tasks. Data analysis must be shared between several expert system modules if knowledge bases are to be kept to a reasonable size. The approach in partitioning the knowledge bases has been to mirror the organization of human experts within the MOC. Modules which support on-line STOCC operations are termed "monitor modules." These modules will be active on a continuous basis and will have capabilities based on the functional responsibilities initially defined for the MOC Operations Engineering console positions. Modules which support off-line operations are, with one exception, termed "diagnostic modules." The exception is the Datamaster module which identifies and subsets telemetry or other data required by the diagnostic modules.

Since the analysis of telemetry from one subsystem is often relevant to another, TALOS modules must communicate. Monitor modules must also have the capability to send a message to a diagnostic module when an anomaly has occurred that needs to be diagnosed. Since the diagnostic modules are not active at all times, messages from the on-line modules will cause the diagnostic module to activate.

Functions supported by the monitors will be essentially identical to those performed by operations personnel, anomaly recognition, uplink management and command execution verification.

Each subsystem monitor module will recognize and provide

ORIGINAL PAGE IS
OF POOR QUALITY

a warning when any vehicle operating constraint, restriction or limitation has been violated or is approaching a violation. When some immediate action should be taken in response to a malfunction the operator will be directed to the correct procedure.

Subsystem modules will generate knowledge of expected events relevant to that subsystem derived from the mission timeline. Modules will, at the time of the expected event, monitor the appropriate telemetry monitors to assure that the on-board stored commands are executing properly to accomplish the expert operation.

A special monitor system, the Operations Controller module, will check real-time commands prior to uplink to ensure that execution of the command by the spacecraft is appropriate for the current vehicle operating mode as determined through analysis of the telemetry. The Operations Controller module will also check uplink loads prior to transmission to ensure the correct load is being uplinked.

Diagnostic modules have two primary functions. First is to reduce the time required to analyze the cause of anomalies and recover from them. The second is to identify anomalous or adverse long term trends in telemetry. Diagnostic subsystem modules will be activated automatically by a monitor module when an anomaly is detected or by a human subsystem engineer. When activated automatically, a module determines what data to analyze and will attempt to diagnose the problem. Results of the analysis, either determination of a probable cause or at minimum elimination of some of the possibilities, will be printed out. Diagnostic modules will also be activated periodically to analyze archived data for anomalous or adverse trends.

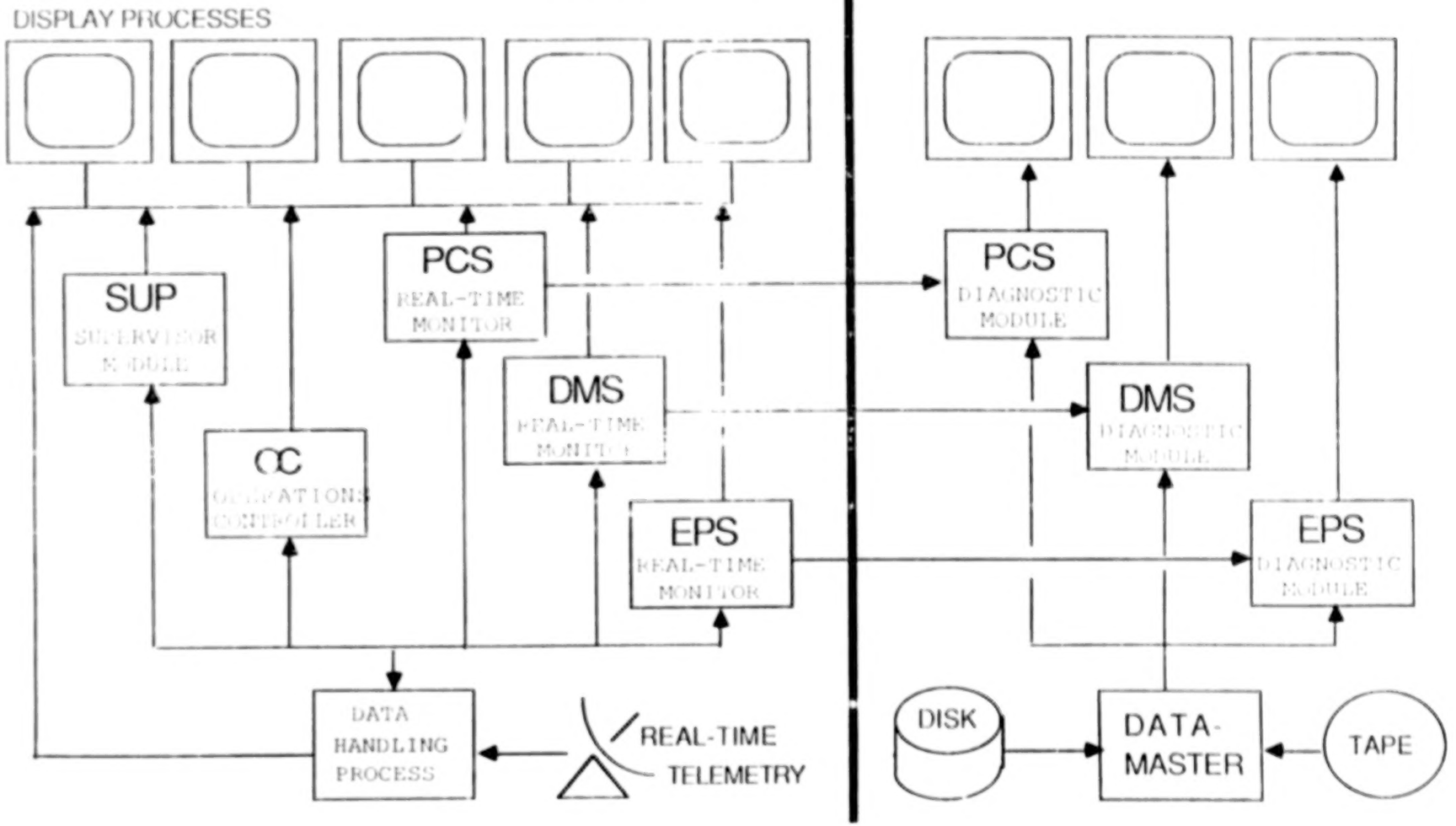
Near term prospects for implementing TALOS on a large scale for the Hubble Space Telescope are uncertain due to budget and schedule constraints. Development work is expected to proceed, however. The general TALOS capabilities are applicable to almost any spacecraft operations control center. The integration of rule-based shells, telemetry data handling utilities, communications capabilities and other spacecraft related utilities that are part of the TALOS architecture can be seen as a generic system for development of highly automated spacecraft control centers.

TALOS:
EXPERT SYSTEM BASED ARCHITECTURE

SPACE TELESCOPE — MISSION OPERATIONS CONTRACTOR

STOCC — L-STAR BASED EXPERT SYSTEMS FOR REAL-TIME VEHICLE MONITORING

SCAR — LES BASED EXPERT SYSTEMS FOR OFF-LINE DIAGNOSIS



N88-24194

Orbital Navigation, Docking, and Obstacle Avoidance As A Form Of Three Dimensional Model-Based Image Understanding

J. Beyer, C. Jacobus, B. Mitchell

Environmental Research Institute of Michigan
P.O. Box 8618, Ann Arbor, MI 48107

ABSTRACT

Range imagery from a laser scanner developed at ERIM can be used to provide sufficient information for docking and obstacle avoidance procedures to be performed automatically. Three dimensional model-based computer vision algorithms in development at ERIM can perform these tasks even with targets which may not be cooperative (that is, objects without special targets or markers to provide unambiguous location points). Roll, Pitch, and Yaw of vehicle can be taken into account as image scanning takes place, so that these can be corrected when the image is converted from egocentric to world coordinates. Other attributes of the sensor, such as the registered reflectance and texture channels, provide additional data sources for algorithm robustness.

1. INTRODUCTION

ERIM has been working towards a laser collision avoidance and spacecraft docking system over the last year. Our interest in this project has been motivated by several key events. We have had a long history of using laser radar techniques for precision robotics control for manipulation of parts in jumbled environments, for navigation and obstacle avoidance for vehicle systems (for both factory and natural environments), and for remote data collection for mapping. We recently (October 1987) were selected by NASA Code IC to be the second Center for Commercial Development of Space Automation and Robotics, and have been tasked with developing the sensing systems component of the Fairchild Space Company Team's Flight Telerobotic Servicer (for NASA Goddard).

To support these activities we began working on the design of a system which will allow imaging and docking with uncooperative spacecraft (or other material). By uncooperative, we mean craft which may not have special reflectors or patterns to facilitate the docking operation. This is in contrast to the work currently underway at Johnson Space Center to demonstrate a system which requires designed targets mounted on the spacecraft (i.e. works with cooperative craft). The concept is based on forming range images from a sensor mounted on a maneuverable spacecraft (Figure 1). These range images are then processed by a model-based image processing system to find distinct object locations which can be used to drive a docking/tracking algorithm.

We discuss how this technology, in two different forms, is applicable to both the obstacle avoidance and the precision inspection (and robot control) problems in a Flight Telerobotic Servicer system, and can be used to validate and analyse large space structures, like

the Space Station.

2. THE THEORY OF DOCKING

Figure 1 shows a schematic form of the basic docking problem. To be able to maneuver to a given spacecraft, it is necessary to be able to measure its orientation and motion relative to the maneuverable platform precisely. In open space (free from other significant gravitational forces), docking can be directly computed from the relative orientation and motion parameters. In an orbital environment, the computations are more complicated, and also require description of the orbital parameters of one of the spacecrafts.

To determine the relative orientation of a craft, the radial measurements from a sensor on the maneuvering platform to three or more known points (P1, P2, P3) on the craft are sufficient. Figure 2 shows three such measurements, R1, R2, and R3. To calculate the satellite coordinate system's unit vectors, U, V, W, as a function of the maneuvering platform's, X, Y, Z (centered at the sensor):

$$U = (R3 - R2)/abs(R3 - R2)$$
$$V = (R1 - R3)/abs(R1 - R3)$$
$$W = U \times V$$

The direction cosine matrix, D, can be converted into any equivalent form, such as pitch, roll, yaw, and is computed as follows:

$$D = \begin{vmatrix} transpose(U) \\ transpose(V) \\ transpose(W) \end{vmatrix}$$

The difference in D from measurement to measurement can be used to estimate changes in orientation (pitch, roll, and yaw) and position for motion estimation. These calculations can be made for any three locations P1, P2, and P3 as long as they are not colinear.

Because of errors in measurement, and because of the possibility of missing measurements (due to loosing identification of P1, P2, P3), it is advisable to feed the the measurements into a Kalman filter. This provides for continuous prediction of spacecraft motion and estimation of probable errors in location.

Table 1. summarizes the design goals for the Johnson Space Center cooperative docking experiment. These have been taken by us as reasonable specifications for the docking sensor, with the exception of the maximum bearing angle, which we believe should be as close to 180 degrees as possible (to allow docking and obstacle avoidance over an entire hemisphere), and pitch/yaw which should allow for +/-180 degrees of rotation.

Table 1. Docking Sensor Requirements and Accuracy

Quantity	Measurement Range	Measurement Accuracy
Range	0-5 mi	1% of range to 0.016 ft
Range Rate	+/-20 ft/s	0.7% of range**1/3 to 0.01 ft/s
Bearing Angle	+/-10 deg	0.05 deg nominally
Bearing Angle Rate	+/-1 deg/s	0.003 deg/s nominally
Pitch and Yaw	+/-45 deg	0.3 deg within 100 ft
Roll	+/-180 deg	0.3 deg within 100 ft
Attitude Rate	+/-6 deg/s	0.01 deg/s within 100 ft

3. THE SENSOR

ERIM's approach to performing the ranging task involves using a three dimensional laser ranger. This technique and the associated technology has been developed and used for obstacle avoidance, robotics applications, and mapping for over ten years. The three dimensional ranger is essentially an optical radar, and is shown in functional form in Figure 3. The laser ranger uses a laser diode operating at 820nm as its source. This diode is amplitude modulated and scanned across the field of view using moving mirrors. The beam is reflected off of the target satellite, and the reflected light is gathered by the receive optics and focused on an avalanche photodiode. The signal from the detector has the same frequency as the laser diode modulation, but displaced in phase. This phase shift is proportional to range.

To measure the phase precisely, a lower frequency waveform is mixed with limiter amplified transmitted and received signals, and the resulting lower frequencies are phase compared digitally. This technique allows range measurement to be made real time, without post processing, except for sensor model correction. By taking measurements in an array, two dimensional images of range data (along with registered reflectance data) are formed. Figures 4 and 5 show representative reflectance and range images of part of a shuttle model (on a non-reflective background to simulated space). Note that range data is indeterminate where reflectance values are zero (black in the reflectance data). Figure 6 shows the range data plotted as a three dimensional surface, after the reflectance channel data is used to gate accurate range values only.

There are two key issues which make the ranging system required for this application somewhat different from those previously designed for vehicle guidance and robotics. The first is the necessity to achieve prescribed ranging accuracies. The basic signal to noise relation is:

$$S/N = Isg^{**}2/(Ishl^{**}2 + Iss^{**}2 + Inep^{**}2)$$

Where Isg is the signal current from the reflected beam, Ishl is the shot noise in the detector due to this signal current, Iss is the shot noise due to solar illumination, and Inep is the detector noise equivalent power current. The range measurement accuracy is limited

by this S/N ratio. Figure 7 shows a set of design curves relating range error to S/N for targets at several ranges, and demonstrated some nominal values for an achievable docking system using this technology.

Another issue is the likely necessity for measuring range over progressively shorter range distances to higher accuracies. The basic phase detection-based ranging scheme is limited to range measurement over fixed ambiguity distances which are determined by the laser modulation frequency, f, (and the speed of light, c):

$$Ra = \text{Ambiguity Interval} = c/2f$$

Measurement over varying ambiguity ranges can either be handled by having programmable modulation sources, or by mixing two frequencies, f(1) and f(2). If the ambiguity intervals of each are:

$$Ra(1) = c/2f(1) \qquad Ra(2) = c/2f(2)$$

Then there will be a beat frequency of f(B) and a corresponding ambiguity interval, Ra(B):

$$Ra(B) = Ra(1)Ra(2)/(Ra(1) - Ra(2))$$

4. MODEL-BASED IMAGE PROCESSING

After acquisition of sufficiently accurate range imagery, the problem of docking becomes one of finding usable (accurately locatable, non-colinear) satellite locating points. In the case of the JSC demonstration system, the problem is simplified by mounting highly reflective, coded targets on the satellite in known configurations. In this way, each reflector point can be located and identified without significant image processing being done.

The problem with this approach is that every object may not have these reflectors mounted on it. For instance, every strut used to construct Space Station space frames will probably not be marked this way. If the strut is dropped by either a construction robot or an EVA astronaut, and retrieval is required, features inherent to the strut will have to be used to determine its position and orientation for docking.

To provide this more general capability ERIM has applied a newly developed model-based vision system, VISTA, developed around the Cytocomputer highspeed flight qualified image processing system, a Symbolics 3600 Lisp machine, and algorithms developed for three dimensional surface identification. VISTA (Figure 8) contains three phases: 1) transformations of images into state-labeled feature maps using conventional image processing (for instance to group co-planar adjacent points), 2) transformation of state-labeled maps into composite symbolic feature maps (in Lisp list form) which describe features (such as lines, surfaces, and vertices) and the relationships between them, and 3) identification by matching prototypical feature-based object models with portions of composite feature symbolic feature maps. VISTA system software is comprised of the

image processing language C4PL, which supports conventional and morphological image processing on the Cytocomputer and the VISTA-WORKBENCH which defines object structures, relations between features, and the VISTA model matching language (and matcher), for defining object (in this case satellite) libraries to drive recognition.

To find and match satellite features, satellite surfaces, and then surface junctions (or edges and vertices) must be found, grouped and coded symbolically. These symbolic quantities can then be matched against pre-coded satellite models in VISTA's library of known objects. The satellite encoded models are currently hand built, but will be build from symbolic input constructed from the range data taken under controlled conditions in the future. The method for aggregating surfaces is also under development, however two methods have been implemented previously. The first finds co-planar points by using local plane estimates, and then the degree of fit of each neighborhood range point to the estimated planes. If the individual range points fit the estimated plane well, then the local area is marked as flat (the flat state). If the points do not fit the plane well, then the areas is marked as discontinuous. If the range points jump discontinuously, then the region is marked as a step discontinuity, otherwise it is left as a roof discontinuity. These states are then used to extract a composite feature map, and matched against models for object segmentation, and then satellite (and therefore, satellite feature point) identification.

A similar surface aggregation algorithm developed by Besl and Jain is being coded to allow generalization from planar surfaces to surfaces which are described by more general polynomials. This enhancement may not be required for satellite docking, because for the large number of planar surface typically found in man constructed objects.

5. APPLICATIONS AND FURTHER WORK

This approach to docking is actually a direct application of three dimensional imaging and model-based image processing to the satellite location and orientation finding problem. This problem is also part of the solution to the collision avoidance problem between controlled multiple robots, between the robots and their workspaces, and between free-flying objects. These problems are all important for the Flight Telerobotic Servicer system, and will be addressed as this project continues.

The automatic generation of VISTA object models is related to the problem of making a consistent CAD database (composed of high-level graphical entities, as opposed to simple collections of range points) from range measurements. We have solved this problem for encoding complex surfaces as meshes, but must do more work to reliably convert these meshes into simpler, and more general graphical entities. This capability allows the accurate capture of solid objects, and can be used as input to solids modelling system to verify accurate object mating, without building and fitting mock-ups. As structures in space become more complex, this capability will be routinely required as

part of subsystem physical checkout.

6. REFERENCES

1. Besl, P. and Jain, R. 1985. Intrinsic and extrinsic surface characteristics. Proceedings of Computer Vision and Pattern Recognition Conference (San Francisco, CA, June 9-13), pp. 226-233.

2. Sampson, R., Swonger, C. W., VanAtta, P. 1984. Real Time 3 Dimensional Image Processing for Robot Applications. SME Robots 8, (Detroit, June).

3. Crimmins, T. R., W. M. Brown, 1985. Image Algebra and Automatic Shape Recognition. IEEE Transactions on Aerospace and Electronic Systems.

4. Jacobus, C., 1986. Autonomous Delivery Vehicle Systems. USPS Advanced Technology Conference, (Washington, D.C., September).

5. Mitchell, B., Gillies, A. 1987. A System for Feature-Model-Based Image Understanding. ERIM Report.

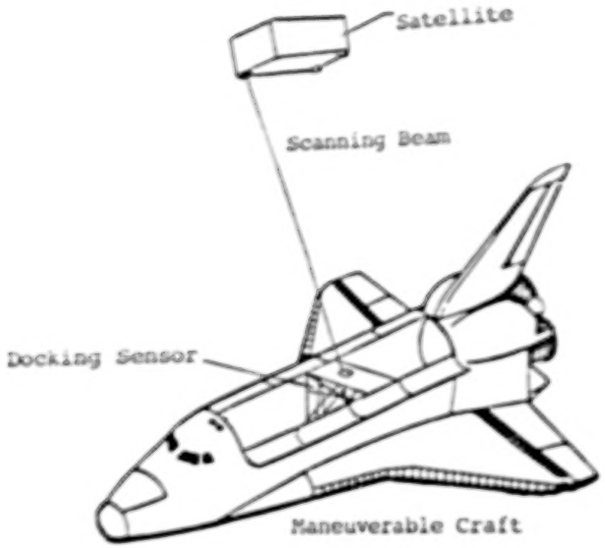


Figure 1. The Laser Docking System Concept

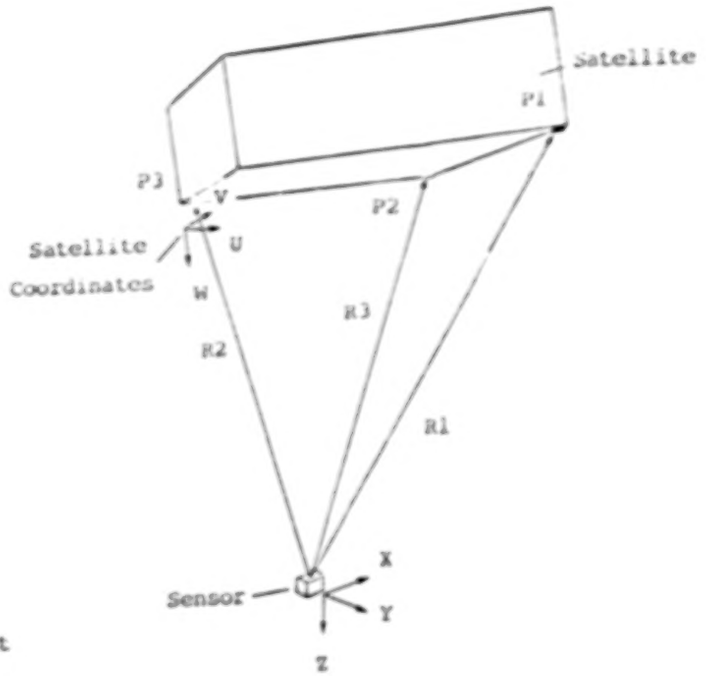


Figure 2. Laser Range Measurements

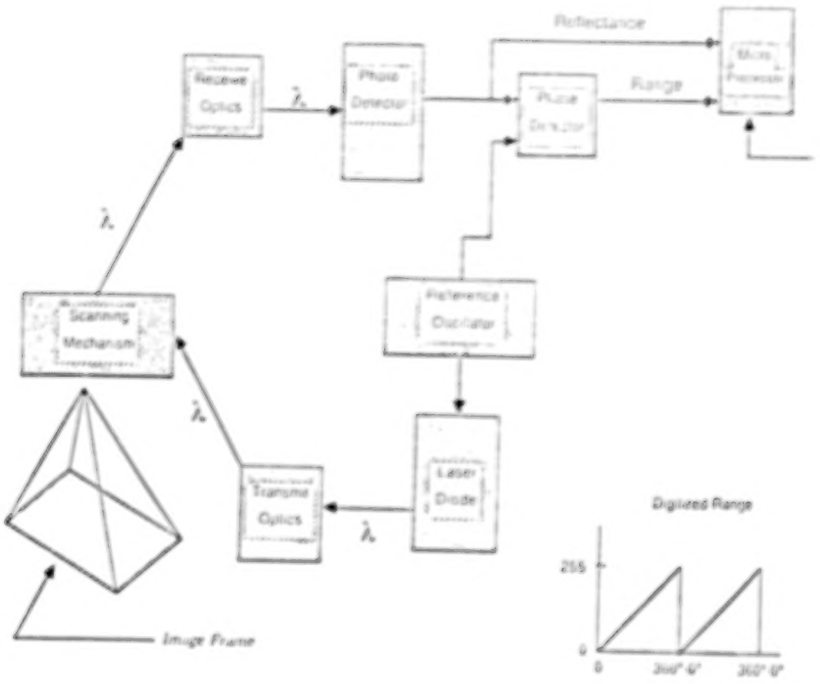


Figure 3. 3D Scanner Block Diagram

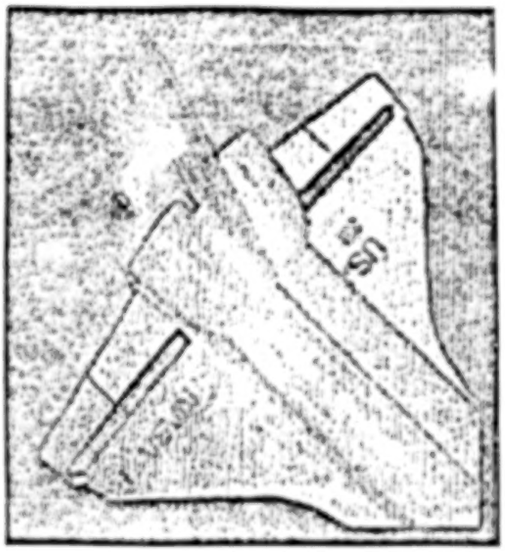


Figure 4. Shuttle Reflectance Image

Figure 5. Shuttle Range Image

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

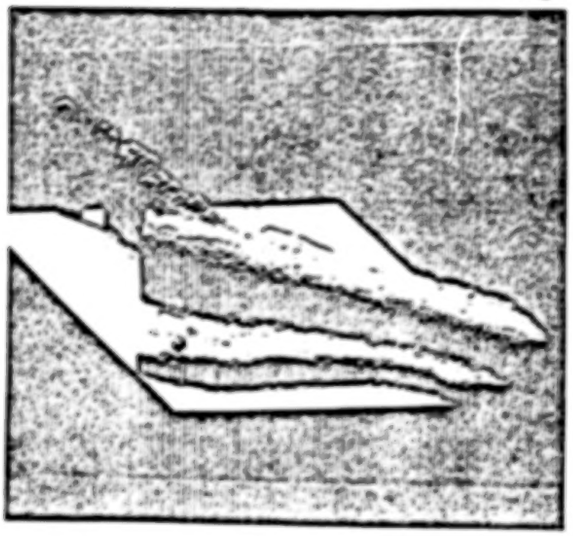


Figure 6. Shuttle Range Image Plotted From Perspective

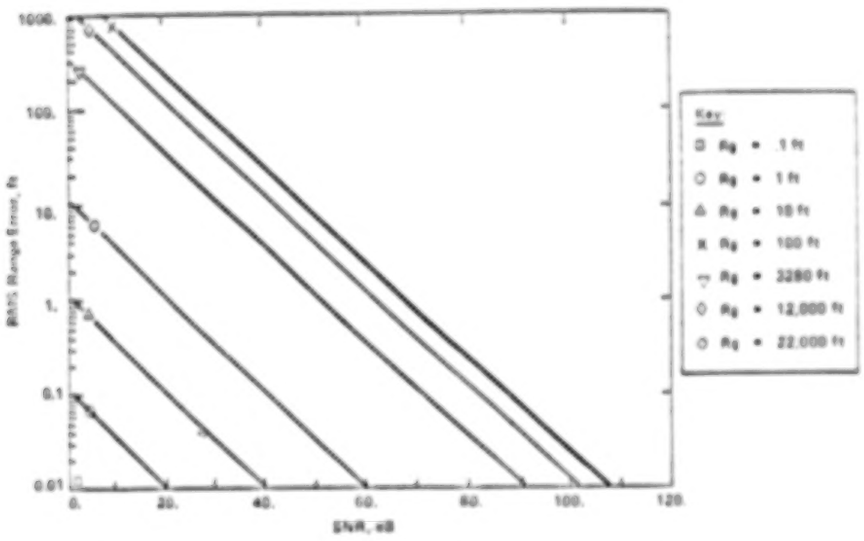


Figure 7. RMS Range Error versus SNR For a Typical Design

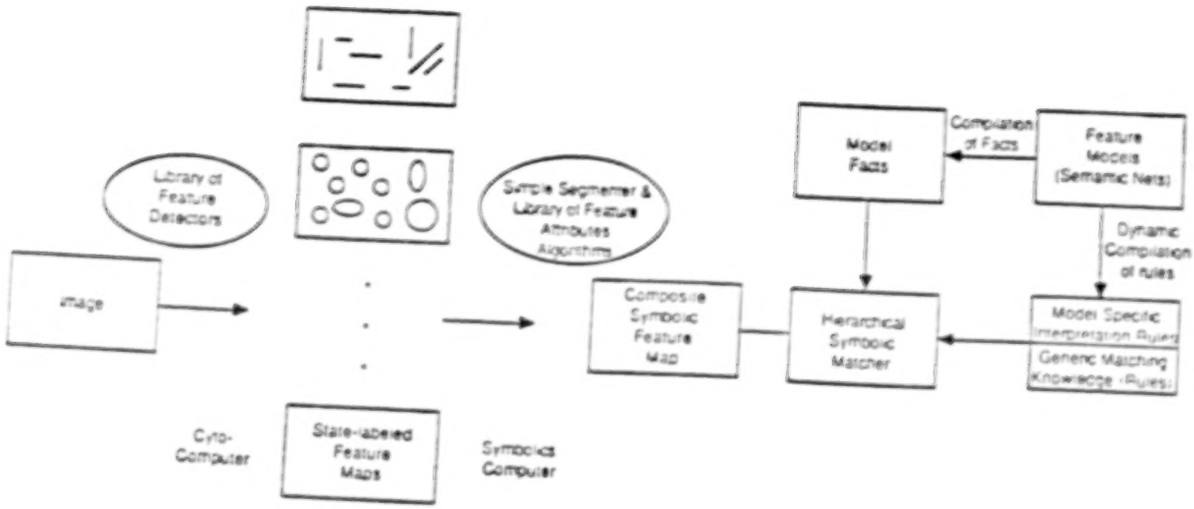


FIGURE 8: VISTA System Overview

N88-24195

Genetic Algorithms for Adaptive Real-Time Control in Space Systems

J. van der Zijp and A. Choudry

Center for Applied Optics
The University of Huntsville in Alabama

ABSTRACT

The U.S. Space Station planned for the 90's will comprise a large number of interacting control systems, a situation which will be too complex for humans to deal with without the aid of knowledge based systems. However, current knowledge based systems have two undesirable aspects. Brittleness: problem-solving performance degrades precipitously when the system is operating outside its intended domain, and adaptation: the ability of the knowledge base to deal with a changing environment.

Learning systems try to alleviate these problems, by organising their own knowledge base. These systems can be divided into two categories: the more traditional rule-based systems on one hand, and neural nets and genetic algorithms on the other. The latter two systems do not operate in the domain, but are 'sandwiched' between a pre- and post processing phase.

Systems based on genetic algorithms are characterized by on-the-fly improvement through a continuous contention of the available rules, while maintaining performance levels as new rules are "tried out." The most salient feature is the employment of techniques from microbiology to conquer the combinatorial explosion involved in induction of the new rules.

INTRODUCTION

Classifier systems are a domain independent way of manipulating knowledge about, say, the characteristics of a real-time control process. The classifier system is embedded into a pre- and post translation phase that relates the knowledge of the classifier system to the environment (the domain). Since it is usually not possible to incorporate all the required knowledge into the system a priori, we want the system to be able to learn.

Genetic Algorithms are used for learning as one way to control the combinatorial explosion associated with generation of plausible new rules. The Genetic Algorithm approach tends to work best when it can be applied to a domain independent knowledge representation.

As important as the generation of new rules is, the evaluation of existing rules is even more influential. Whithout it, the system will not be able to attain better performance.

CLASSIFIER SYSTEMS

As stated, a classifier system is 'sandwiched' between a separate pre- and post processing phase. The preprocessing phase will pick up the relevant information from the environment and encode this into the internal representation used by the classifier system, called a message. Likewise, the postprocessing phase converts back from the internal representation to a meaningful output signal, such as an actuator, a horn signal or whatever.

We will not discus the pre- and postprocessing phases in greater detail here, although for applications of classifier systems in real-world situations these are of course tantamount to good performance.

A classifier system comprises two parts, a message buffer and a rule base. Communication with the world outside takes place exclusively by placement or removal of messages into or from the message buffer. Also acting on the message buffer is the rule base itself. In each classification step, all messages are matched to all rules, and the matching pairs (rule,message) are allowed to write a new message into the buffer.

MESSAGE REPRESENTATION

A message is internally represented as a string of K symbols from {0,1} i.e. a binary representation. Each symbol denotes some aspect or attribute of the environment. As will be seen, this representation is very suited for genetic algorithms.

Rules in the rule base consist of two parts: a condition part and an action part. The condition part is represented as a number of strings of length K, over the alphabet {0,1,#}. The new symbol # is used here to match ANY symbol in the corresponding position in the message. For instance, condition 0101#01# will match messages 01010010, 01010011, 01011010 and 01011011. Each condition can be either asserted or negated.

The rule is said to MATCH if ALL the conditions match some message in the buffer (i.e. a match does NOT mean that one message satisfies all conditions, but rather that there is a satisfying message in the buffer for each condition). The message matching the first condition is arbitrarily called the matching message.

The action part is also represented as a string over {0,1,#}. In this case, however, the symbol # is used to denote the corresponding symbol of the message matching this rule. In other words, should a rule with action part 0101#01# match a message 11011000, then the message placed into the buffer by this rule will be 01011010.

BASIC EXECUTION CYCLE OF A CLASSIFIER SYSTEM

A classifier system goes through the following steps for each classification cycle. For legibility, this is here presented in (pseudo-) pascal style:

```
repeat
    input_messages_&_place_them_into_buffer;
    for r := 1 to no_of_rules do begin
        condition := first_condition_of_rule_r;
        message[condition] := 0;
        repeat
            message[condition] := message[condition]+1;
            if message[condition] < no_of_messages then begin
            if match(condition,message[condition]) then begin
                if condition=last_condition_of_rule_r then
                    record_match(r,condition,message[condition])
                else begin
                    condition := condition+1;
                    message[condition] := 0;
                end;
            end;
            end else
                condition := condition-1;
        until condition<0;
    end;
    for each_recorded_match do place_action_message_in_new_buffer;
    update_strength_of_matching_rules;
    replace_current_message_buffer_by_new_buffer;
    remove_messages_from_buffer_&_output_them;
until hell_freezes_over;
```

It should be noted that not all messages are recognized by the output interface as messages to be removed; only those marked as such will be output. Also note that there may be several different assignments of messages to one rule; these are all recorded.

RULE COMPETITION

Since the number of pairs (rule,message) can be extraordinarily large (with N rules, an average of C conditions per rule, and M messages there can be as much as N*C*M pairs), the number of messages that are eventually placed in the buffer are reduced by letting the matching rules enter a competition. Each matching pair produces a number called the BID, which determines the probability that the pair is allowed to place its message into the buffer. The bid is calculated as follows:

bid=factor*support(messages)*strength(rule)*specificity(rule)

where:

factor = attenuation factor;

support = sum of the bids of all messages satisfying a condition of the rule;

strength = the current "successfulness" of the rule;

specificity = 1-(number of #'s in condition)/(length of condition).

The bid computed above is an exact quantity; were we to use this quatity directly, only the strongest few rules would ever be selected in the competition and new, potentially better rules, would never have the opportunity to demonstrate their worth. Therefore, the bid used in the competition is some stochastic function of the value computed above (at the moment, normally distributed around the computed bid).

RULE ACCREDITATION

In order to allow new (low-strength) rules to acquire more strength in those situations where they are applicable, some mechanism must be devised that "gives credit to whom credit is due." That is, the well working new rules must be rewarded for providing good answers, whereas those that do not do well must sink into oblivion.

Such a mechanism has been devised by Holland. It is called the "bucket brigade." The bucket brigade algorithm works by applying the following adjustments to a rule's strength at each classification cycle:

strength(r,t+1) = strength(r,t) - bid(r,t)

strength(i,t+1) = strength(i,t) + bid(r,t)/card({R}) for i in {R}

where {R} is the set of rules responsible for the match of rule r.

In words, each rule which wins the bidding contest gets its strength reduced by the amount of its bid; each of the rules that put a message into the buffer that eventually matched one of the rule's conditions are rewarded for this by having their strength increased; they share the bid among each other.

Eventually, the rule placing a message into the buffer that is consumed by the output interface gets a reward directly by the environment.

In this way, benign "mutations" are able to gain strength. Chains of non-productive rules are eventually broken down because they will receive no credit from the environment.

RESEARCH GOALS

Our interest in classifier systems falls into two categories. On the one hand, we are concerned with the use of classifier systems for real-time adaptive control applications, and the implications of on-the-fly learning for robustness. On the other hand, we envisage some improvements to the mechanism of the classifier system itself, e.g. the guiding of the 'mutation' process by meta-rules, other kinds of genetic operators etc. Also, we want to study the effects of 'teaching' to the speed of adaptation.

REFERENCES

Holland, J. H. "Adaptation in Natural and Artificial Systems," The Univerity of Michigan Press, Ann Arbor.

Holland, J. H. "Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-Based Systems," Machine Learning, Vol II.

Holland, J.H., Holyoak, e.a. "Introduction to Processing of Interface Learning and Discovery," 1986 MIT Press.

Hilliard, M. R., Liepins, "A Classifier Based System for Discovering Scheduling Heuristics," 2nd Int. Conf. on Genetic Algorithms, July 1897.

Goldberg, D.E. Segrest, P., "Finite Markov Analysis of Genetic Algorithms," 2nd Int. Conf. on Genetic Algorithms.

N88-24196

AUTOMATIC PROGRAM GENERATION FROM SPECIFICATIONS USING PROLOG

ALEX PELIN
School of Computer Science
Florida International University
Miami, Florida 33199

PAUL MORROW
AFWL / SCR
Kirtland AFB, NM 87117

Abstract

This paper describes an automatic program generator which creates PROLOG programs from input/output specifications. The generator takes as input descriptions of the input and output data types, a set of tests, a set of transformations and the input/output relation. Abstract data types are used as models for data. They are defined as sets of terms satisfying a system of equations. The tests, the transformations and the input/output relation are also specified by equations.

The program generator creates a PROLOG program which takes as input a data item, item1, of the input data type and outputs a data item, item2, of the output data type such that the input/output relation is satisfied by the pair item1,item2. In building the program the generator uses only the tests and the transformations given as input. The program generator was writen in PROLOG. It was able to generate correct PROLOG programs for sorting lists with and without eliminating duplicate elements. The system contains specifications of the abstract data types natural number, boolean, list and array. The system can be used in two modes: in the first mode the user defines his/her own data types and in the second mode he/she uses the definitions that are already in the system library. A user interface is being constructed for the second class of users.

The paper also presents several methods for validating the input/output specifications. Some of them were implemented in the system. Descriptions of the heuristics employed by the program synthesizer are also included. Finally, the paper compares this system with the approaches taken by other researchers in automatic program generation : Prywes, Manna and Dershovitz.

Introduction

This paper describes a project whose goal is to create a knowledge base for automatic program generation from input/output specifications . By its scope the project falls into the category of program synthesis. The goal of program synthesis is two-fold : to develop specification languages in which one can describe programming tasks and to develop heuristics which translate these specifications into high level language code.

The system described in this paper took as input the following items:

1. A description of the input data type;
2. A description of the output data type;
3. A description of the input/output relation;
4. A set of transformations;

PRECEDING PAGE BLANK NOT FILMED

PAGE 52

5. A set of tests.

Figure 1

Based on the input data, the system tries to generate a program P which takes as input an item i of the input data type and outputs an item o of the output data type such that the input/output relation holds for the pair i,o. The program P is constructed by using only the tests and the transformations given as input to the program synthesizer. For example, the problem of sorting a list of natural numbers in increasing order is specified by the 5 items shown below.

1. The input data type is list of natural numbers;
2. The output data type is list of natural numbers sorted in increasing order;
3. The input/output relation is standard;
4. The set of transformations is inversion of consecutive elements;
5. The set of tests consists of the predicate SORTED, which checks if a list of natural numbers is sorted in increasing order, and the predicate less which checks if two consecutive elements in the list are in the right order,i.e. the first element is less than or equal to the second element.

Figure 2

Two data items i and o satisfy a standard input/output relation if o is obtained from i by applying a sequence of transformations. Each transformation in the sequence must be a member of the set listed in item 4 of figure 1. The tests listed in item 5 can be used to guide the transformations. The program synthesizer was able to generate a correct PROLOG program for the specifications shown in figure 2.

The paper is organized as follows: issues pertaining to the problem of validating program specifications are presented in section 2, heuristics used to generate PROLOG code are described in section 3 and the user interface is presented in section 4.

2. Validation of Specifications

The system uses abstract data types ([3]) as models for data. A data type is a set of terms, freely generated by some operations, which satisfies a set of equations (conditional equations). For example, the data type list of natural numbers consists of the data types :natural number, list of natural number and boolean. The terms of the data type are generated by the productions (rules) shown in figure 3.

```
natural ---> 0
natural ---> successor(natural)
list    ---> []
list    ---> [ natural | list]
boolean ---> True
boolean ---> False
boolean ---> natural <= natural
```

Figure 3

The variable 'natural' generates all the natural numbers, the variable 'list' generates all lists of natural numbers and the variable boolean produces all terms of type boolean. The operator 'successor' is the successor operator on the set of natural numbers. The semantics of the data type is given by the set of (conditional) equations presented below.

1. n <= n = True
2. n <= m = True ---> successor(m) <= n = False
3. n <= m = True ---> n <= successor (m) = True

Figure 4

The equations given in figure 4 describe the relation <= on the set of natural numbers. Since the user specifies the data types that are used by the system, it is fundamental that the system validates these specifications. An example of a validation problem is the following : for all natural numbers m and n, either m <= n = True or n <= m = False. In this system, data type validation questions become theorem proving problems. A data type validates a property if that property is a theorem produced by the system of axioms that define the data type. There are several methods that can be used to accomplish this goal. One can use first order logic, term rewriting systems and induction. The system described in this paper uses term rewriting systems as the main tool for validating data types. The system has a Knuth-Bendix completion procedure ([7]) that can be used to generate a complete set of reductions for a system of equations. At present the completion procedure operates only with pure equations but there are ways of extending it to conditional equations([5]). Several experiments were made with the interactive theorem prover ITP. There were many problems with the use of ITP for validating data type specifications. This theorem prover, like all theorem provers based on resolution, tends to generate a tremendous amount of useless clauses. It is therefore imperative to develop heuristics that eliminate clauses that are irrelevant to the proof of the question to be validated. The other problem is that the relation between first order logic and PROLOG is a complicated one. The translation of a set of first order logic sentences into a set of Prolog clauses is not easy. The third problem with using first order logic to validate questions in this system is that, since the data types are inductively defined, theorems that require induction cannot be proved. For example the theorem which states that for all natural numbers m, m <= successor(m) requires induction. Since the progress in automating induction has been slow, the system does not use direct inductive methods.

3. Heuristics Used by the System

So far the system can deal with problems in which the output data type is a subset of the input data type. Sorting problems fall in this category. The data type list of natural numbers sorted in increasing order can be defined as the subset of the data type list which satisfies the predicate SORTED described below:

1. SORTED([])
2. SORTED([m])
3. m <= n = True ---> SORTED([m |[n |list]]) = SORTED([n |list])
4. m <= n = False ---> -SORTED([m|[n|list]])

Figure 5

In figure 5 - stands for negation. The condition m <= n = True, m <= n = False correspond, respectively, to tests -less(n,m) and less(n,m) shown on line 5 of figure 2.For the class of problems in which the output data type is a subset of the input data type and the input/output relation is standard, the system focuses on the predicate that defines the output data type. For the problem described in figure 2, the system transforms the third clause of the specification shown in figure 5 into two clauses:

```
5. m <= n = True , SORTED([n|list]) ---> SORTED([m|[n|list]])
6. m <= n = True ,-SORTED([n|list]) --->-SORTED([m|[n|list]])
```

Figure 6

Then the system employs a method which focuses on the negative clauses from figures 5 and 6. If SORTED(list) is false then the clause -SORTED(list) must occur either at the right of the ---> sign in clause 4 of figure 5 or at the right of the ---> sign in clause 6 of figure 6. The system assigns higher priority to clause 6 than to clause 4. In general clauses that contain recursive calls have higher priority than those that do not have them. If -SORTED(list) is derived from rule 4 of figure 5 then the system establishes the negation of the condition m <= n = False as its goal. In this case the system looks for a transformation, or a sequence of transformations , T_list such that: if -SORTED(list) is obtained from rule 4 then the precondition of rule 4 does not apply to T_list(list). In this particular case the system looks for a transformation that carries the argument [m|[n|list]] of the rule 4 of figure 5 into a list which is sorted, or it has length less than the original argument, or it is of the form [p|[q|list']] and p <= q = False does not hold. The system has a generate and test algorithm for finding T_list. The states are pairs of the type <Term, Tramsformation>, where Tranformation is a sequence of transformations that brings the original argument to the list Term. It uses various guiding functions for searching the state space. The method was tested using the sets of transformations {EXCHANGE} and {EXCHANGE,REDUCE}.The equations for EXCHANGE and REDUCE can be found in figure 7.

```
1. EXCHANGE([m|[n|list]] = [n |[m |list]]
2. REDUCE([m|[m|list]) = [m |list]
```

Figure 7

The system produced correct answers in both cases. The predicates must be defined in a hierarchical manner. This means that test1 can be defined as a function of test2, but test2 cannot also be defined as a function of test1.

4. The User Interface

The system can be used in two ways. In the first mode the user defines his own data types, tests, transformations and input/output relation. In the second mode the user employs the definitions that are already in the system library. The system has an interface that allows the user to enter commands in English. The user can enter statements like the ones shown in figure 8.

```
1. X is a real array
```

2. 3 is the size of X
3. Y is X reversed
4. Display Y

Figure 8

These types of instructions allow users to employ the system like a calculator. Work is under way to enrich the interface to the point where it can accept atatements like the ones shovn in figure 2 . In this case the user will receive from the system the program that accomplishes the task described in English.

5. Conclusions

Automatic program generation is an important problem in automating the software development cycle ([1],[4]). It can be used do develop program modules from task specifications. Defining specifications languages is a difficult job ([6],[10]). Validating the specifications is more difficult. The system uses equations for specifications. The system MODEL, developed by the group led by N. Pryves ([9]), also uses equations for specifications. Dershovitz ([2]) employs term writing systems to synthesize programs. In the system presented in this paper, term rewriting systems are used for validating specifications and as an intermediate step in translating equations into PROLOG clauses. Dershovitz uses Pascal to carry out program synthesis. PROLOG seems better suited for implementing heuristics. The programs generated by the system are quite different than those which are constructed through informal means. This fact is mentioned by Manna ([8]).

Acknovledgement

This work was supported by the Air Force Office of Scientific Research/AFSC under contract F496-C-0013.

Bibliography

1. Boehm, B. : 'Improving Software Productivity', Computer, Vol. 20, No. 9, September, 1987, pp. 43-57.
2. Dershovitz, N. : 'Synthesis by Completion', proceedings of IJCAI-85, Morgan Kaufmann, 1985, pp. 208-214.
3. Ehrig, H. and Mahr, B. : Fundamentals of Algebraic Specification 1, Springer Verlag, 1985.
4. Frenkel, K. :'Towards Automating the Softvare-Development Cycle', CACM, Vol. 28, No. 6, June, 1985, pp. 578-591.
5. Ganzinger, H. : 'A Completion Procedure for Conditional Equations', University of Dortmund Technical Report 234, October,1987.
6. Hoare, C. :'An Overview of Some Formal Methods for Program Design',Computer, Vol. 20, No. 9, September, 1987, pp. 85-91.
7. Knuth, D. and Bendix, P. : 'Simple Word Problems in Universal Algebras', Computational Problems in Abstract Algebra, Pergamon Press, 1970, pp.80-149.
8. Manna, Z. and Waldinger, R. :'The Origin of the Binary Search Paradigm', proceedings of IJCAI-85, Morgan Kaufmann, 1985, pp. 222-224.
9. Pryves, N. , Shi, Y. ,Szymansky, B. and Tseng, T. :'Supersystem Programming with Model' , Computer, Vol. 19, No. 2, February, 1986, pp. 50-60.
10. Roman, G. : 'A Taxonomy of Current Issues in Requirements Engineering', Computer, Vol. 18, No. 4, April, 1985, pp. 14-22.

N88-24197

TES - A Modular Systems Approach to Expert System Development for Real-Time Space Applications

Ralph Cacace
Brenda England
UNITED TECHNOLOGIES CORPORATION
Hamilton Standard Division
One Hamilton Road
Windsor Locks, CT 06096

Abstract

A major goal of the Space Station Era is to reduce reliance on support from ground based experts. The development of software programs using expert systems technology is one means of reaching this goal without requiring crew members to become intimately familiar with the many complex spacecraft subsystems. Development of an expert systems program requires a validation of the software with actual flight hardware. By combining accurate hardware and software modelling techniques with a modular systems approach to expert systems development, the validation of these software programs can be successfully completed with minimum risk and effort. The TIMES Expert System (TES) is an application that monitors and evaluates real-time data to perform fault detection and fault isolation tasks as they would otherwise be carried out by a knowledgeable designer. This paper discusses (1) the development process and primary features of TES, (2) a modular systems approach and (3) lessons learned.

Introduction

The Thermoelectric Integrated Membrane Evaporation Subsystem (TIMES) is a spacecraft life support system designed and produced by *Hamilton Standard* . The TIMES Expert System (TES) is a prototype developed as a first step toward capturing in-house diagnostic expertise on this subsystem. The primary goals of the prototype development effort were to (1) investigate expert systems as space-based tools for reducing reliance on ground based support and (2) explore a modular systems approach to developing real-time expert systems. To address these goals, TES was developed as a fault detection and fault isolation expert system to monitor real-time TIMES data. The TIMES Expert System was created under NASA/Johnson Space Flight Center contract number NAS 9-17209.

What is TIMES?

TIMES is a spacecraft waste water processor that employs vacuum distillation, thermoelectric heat pumping, and membrane separation to reclaim high quality product water [2]. A system controller contains the electrical and software logic necessary for process

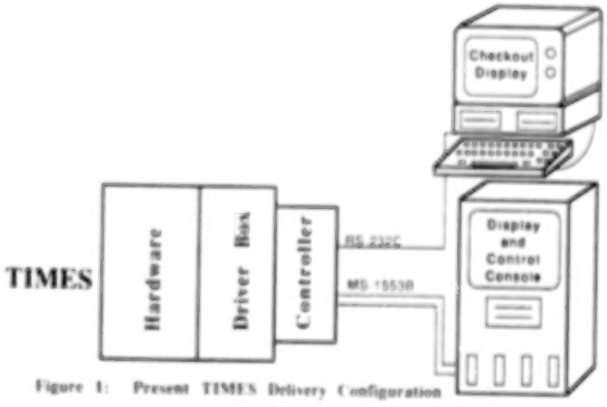


Figure 1: Present TIMES Delivery Configuration

PRECEDING PAGE BLANK NOT FILMED

control, instrumentation and critical fault detection. In addition to data provided to a MS-1553B bus, sensor readings and performance calculations are made available by the controller via RS-232C for use by data checkout and recording systems. An electrical driver box standardizes all signals between the hardware and the controller. The TIMES, along with numerous other life support subsystems, is operated and visually monitored from a central Display and Control Console (DCC) via a MS-1553B bus (see Figure 1).

Primary TES Features

The TIMES Expert System was developed using Knowledge Engineering Environment (KEE) and Common Lisp on a *Symbolics* computer. Approximately 45 potential subsystem and sensor problems have been addressed as both independent and concurrent failures. The TIMES controller relies on threshold violations to provide limited error detection and will shut the subsystem down if a critical problem is suspected. TES enhances controller capabilities by operating as an early warning system, providing additional fault tolerance and acting as an extended problem explanation tool for the astronaut.

Combining the graphics features of KEE and Common Lisp has given TES versatile display capabilities. Some examples are dynamic flows, valve positions, and sensor readings presented on an operational schematic of the TIMES. In addition to these features, the operator also has the ability to customize the monitoring environment by replacing some or all numerical displays with dials and to view real-time historical trend plots of expected and actual values for all sensor and performance data.

A Modular System Design

When real-time prototype hardware output results in inputs to an expert system, the following key issues must be addressed:

1. Limited availability of system hardware for expert system testing and verification due to:
   - A limited number of each prototype system is usually developed and available.
   - Prototype systems are often dedicated to rigorous test schedules and numerous design changes.
   - System testing often has priority over expert system testing.
   - It is costly and difficult to move hardware from one site to another for demonstration or testing purposes.
2. Actual insertion of faults into prototype hardware is unrealistic in many cases because:
   - Altering a system component to introduce a fault can be costly and sometimes hazardous.
   - Many system sensors and components are not physically accessible for alteration.

These issues may be addressed by taking a modular approach to system design. The modular approach involves the development of an accurate

software simulation of the hardware (See Figure 2). The closer the model is to duplicating the hardware's data output in terms of content, format and protocol, the more valuable it will be when addressing the above issues.

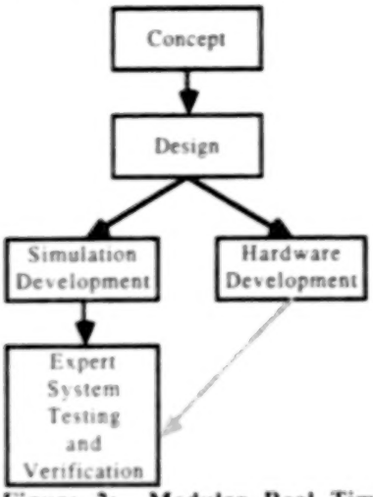


Figure 2: Modular Real Time Testing

Modularity has been applied to the TES program by using a thermodynamic model of TIMES as the expert system test data source. The TIMES model was developed using Quick BASIC on an IBM PC. The RS-232C output from this model realistically duplicates the RS-232C test data output from the TIMES controller in content, format and protocol. This data supplies TES with the sensor, control and performance information it needs to monitor the TIMES (model).

The TIMES model can be operated as a stand alone system or in conjunction with the TIMES Expert System. The model's graphics interface allow the user to visually monitor simulated normal TIMES operation, or to insert faults and observe the effects on subsystem operation independent of the TIMES Expert System. This allows for independent model verification using TIMES data and makes the model itself a valuable tool in understanding detailed TIMES performance and fault characteristics. The model, with its interface (Figure 3), effectively replaces the TIMES hardware and the DCC (Figure 1) for testing and demonstration of TES.

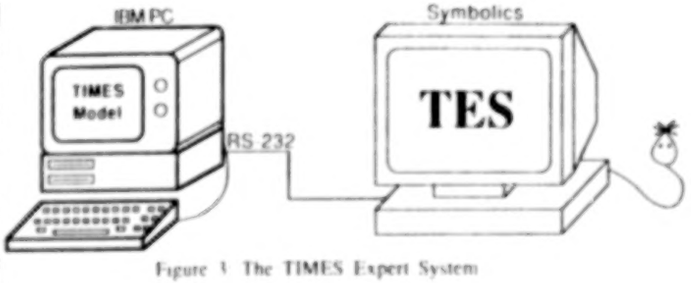


Figure 3 The TIMES Expert System

The modular approach taken in the TES design has helped to reduce development costs by providing a more flexible and accessible testing and verification platform. For example, TIMES operates on an approximate 24 hour cycle and fault symptoms often vary as a function of the cycle duration. To test the TES fault detection capabilities and insure that cyclic data is being handled successfully, testing must be conducted at numerous cycle points. Tests must be repeated a number of times because expert system evaluation is an iterative process [4] . These test constraints would demand valuable subsystem time and may require costly repair or replacement of components damaged during testing or additional hardware to simulate failures. By using a software model as the test bed, an open circuit in a thermoelectric module or a separator motor magnetic drive decoupling can be generated repeatedly. Simulation rates can be adjusted to decrease test time. In these ways, the TIMES model facilitates testing without damaging subsystem hardware or interrupting hardware testing.

A Modular Diagnostic Approach

Remaining consistent with a modular systems approach, TES's diagnostic content is also structured modularly. Application of verbal reporting techniques [3] during the formalization phase [4] of TES's development revealed a distinctly two level diagnostic approach used by the experts. High level performance parameters provide the first clues to a subsystem performance problem. Detailed diagnostics are only employed when such a problem is detected. Because it is the function of the TIMES to efficiently reclaim high quality product water, there are three primary indicators of degraded performance: (1) low water production rate, (2) poor product water quality, and (3) high power consumption. The first two areas have been addressed by the TES prototype. This approach has been captured in a modular fashion modelled after that of the experts allowing for rapid expert system performance, and facilitating future expansion to include other systems [1], [5].

A front end processor in TES monitors TIMES performance parameters using trend analysis to watch for significant indication of a problem. During normal operation, the only other functions performed by TES are the storing of trend data, real-time display update and periodic sensor checkout to flag drifts or inconsistent readings. Only when subsystem performance appears to have deteriorated does TES perform detailed diagnosis to isolate the problem and warn the astronaut before shutdown thresholds are exceeded. (Figure 4)

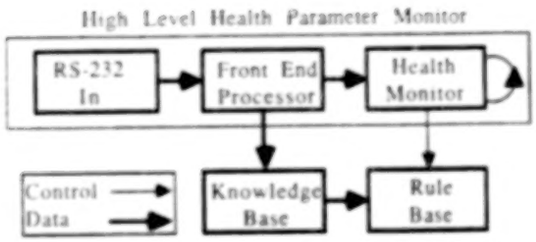


Figure 4: TES Functional Modularity

The rules in the TIMES expert system have been divided into four major categories: (1) General Health rules, (2) Water Production Rate rules, (3) Water Quality rules and (4) Sensor Health rules. Division of the rule base into problem areas has further enabled rapid fault diagnosis. For instance, if a low water production rate were detected, TES would reason over rule types (1) and (2) drawing on the current knowledge base and trend information to diagnose the problem. A separate Sensor Health rule base has increased TES credibility by flagging unreliable sensors so that they are not relied on in future reasoning, and by differentiating between sensor drifts or failures and actual subsystem health problems (i.e. high temperature).

Conclusions

Using the TIMES model as an input to the TIMES Expert System provides a flexible and portable means of addressing the availability, cost, and time issues associated with developing an expert system to monitor real-time hardware data. It also increases TES's diagnostic credibility by separating fault insertion and fault detection. The operator can observe TIMES sensor readings and performance indicators on the model to reach his own conclusions, while the expert system provides the expert diagnosis.

TES's, functional modularity, patterned after the expert's own diagnostic approach, increases its monitoring efficiency without limiting

ORIGINAL PAGE IS
OF POOR QUALITY

future growth. The front end processor makes TES an efficient, high level health monitor until detailed anomaly analysis is needed. The front end processor also supplies TES (and hence the operator) with a trend history of all sensors and performance parameters. Detailed trend analysis on this data supplies the TES reasoning system with the information that the expert uses by either visually examining trend plots or performing detailed analysis of historical data.

Independent sensor monitoring increases TES's diagnostic capability by flagging unreliable sensors thus excluding them from use in future reasoning and by distinguishing between sensor drift or failure and actual performance problems.

Future TES expansion areas could include: (1) further development of the front end processor to address more complex data characteristics, (2) a detailed feasibility study that would address the modular system design to multiple systems, (3) connection to TIMES hardware, and (4) further investigation of multiple faults.

References

[1] A. K. Colling Jr., T. A. Nalette, et.al., "Development Status of Regenerable Solid Amine CO2 Control Systems," SAE Technical Paper Series, Proceedings of the 15 Intersociety Conf on Environmental Systems, San Francisco, CA, July 15-17, 1985.

[2] G. F. Dehner and D. F. Price, "Thermoelectric Integrated Membrane Evaporation subsystem Testing," SAE Technical Paper Series, Proceedings of the 17th Intl. Conf. on Environmental Systems, Seattle, WA, July 13-15, 1987.

[3] A. H. Silva and D. C. Regan, "Using Cognitive Psychology Techniques for Knowledge Acquisition," IEEE Trans. on Systems, Man and Cybernetics, Atlanta, GA, October 1986.

[4] M. Stefik, J. Aikins et. al., "Basic Concepts for Building Expert Systems," in F. Hayes-roth, D. Waterman and D. Lenat (eds), Building Expert Systems, Addison-Wesley Publishing, Reeding, MA, 1983.

[5] C. E. Verostko and R. K. Forsythe, "A Study of Sabatier Reactor Operation in Zero 'G' ," SAE Technical Paper Series, Proceedings of the 14th Intersociety Conference on Environmental Systems, San Diego, CA, July 16-19, 1984.

KEE is a registered trademark of Intellicorp Inc.
Quick Basic is a registered trademark of Microsoft Inc.

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
NASA CP-2492		

4. TITLE AND SUBTITLE	5. REPORT DATE
Third Conference on Artificial Intelligence for Space Applications - Part II	June 1988
	6. PERFORMING ORGANIZATION CODE

7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT #
Compiled by J. S. Denton, M. S. Freeman, M. Vereen	

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. WORK UNIT NO.
George C. Marshall Space Flight Center Marshall Space Flight Center, Alabama 35812	M-576
	11. CONTRACT OR GRANT NO.

12. SPONSORING AGENCY NAME AND ADDRESS	13. TYPE OF REPORT & PERIOD COVERED
National Aeronautics and Space Administration Washington, D.C. 20546	Conference Publication
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES
Conference Coordinator - Thomas Dollman, Information and Electronic Systems Lab, Marshall Space Flight Center
Co-sponsored by The University of Alabama in Huntsville

16. ABSTRACT

Proceedings of a conference held at Huntsville, Alabama, on November 2 and 3, 1987. This Third Conference on Artificial Intelligence for Space Applications brings together a diversity of scientific and engineering work and is intended to provide an opportunity for those who employ AI methods in space applications to identify common goals and to discuss issues of general interest in the AI community.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Artificial Intelligence Computer Vision Design Data Capture Robotics Space Station Automation	Unclassified/Unlimited Subject Category: 61

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	69	A04

MSFC - Form 3293 (Rev. December 1972)

For sale by National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1988

END

DATE

FILMED

AUG 23 1988